08-01-00 A

JC878 U.S. PTO 07/31/00

EXPRESS MAIL LABEL NO. EM598711275US

Attorney Docket No. 74218/05085

JC857 U.S. PTO 09/629370 07/31/00

Box Patent Application
Assistant Commissioner for Patents
Washington, DC 20231

NEW APPLICATION TRANSMITTAL

Transmitted herewith for filing is the patent application of

Inventor(s) : Bruce T. Petro, Andrew Cohen and Jason Sulak

For (title) : ON-LINE SYSTEM FOR CREATING A PRINTABLE PRODUCT

1. Type of Application

This new application is for a(n):

(X) Original (nonprovisional)
( ) Continuation
( ) Continuation-in-part (CIP)
( ) Divisional
( ) Design
( ) Plant

NOTE: If continuation, CIP or divisional, then complete section 2.

CERTIFICATION UNDER 37 C.F.R. 1.10*
(Express Mail label number is mandatory.)
(Express Mail certification is optional.)

I hereby certify that this New Application Transmittal and the documents referred to as attached therein are being deposited with the United States Postal Service on this date July 31, 2000, in an envelope as "Express Mail Post Office to Addressee," mailing Label Number EM598711275US addressed to the: Assistant Commissioner for Patents, Washington, D.C. 20231.

Valerie A. Milam

[illegible]

Signature of person mailing paper

WARNING: *Certificate of mailing (first class) or facsimile transmission procedures of 37 C.F.R. 1.8 cannot be used to obtain a date of mailing or transmission for this correspondence.*

****WARNING:*** *Each paper or fee filed by "Express Mail"* ***must*** *have the number of the "Express Mail" mailing label placed thereon prior to mailing. 37 C.F.R. 1.10(b).*

2. Benefit of Prior U.S. Application(s) (35 U.S.C. 119(e), 120, or 121)

NOTE: If the new application being transmitted is a continuation, CIP or divisional, of a parent case, or where the parent case is an International Application which designated the U.S., or the benefit of a prior **provisional** application is claimed, then check the following item and complete section as follows.

() The new application being transmitted claims the benefit of prior U.S. application(s).

2.1 Relate Back

WARNING: If an application claims the benefit of the filing date of an earlier filed application under 35 U.S.C. 120, 121 or 365(c), the 20-year term of that application will be based upon the filing date of the earliest U.S. application that the application makes reference to under 35 U.S.C. 120, 121 or 365(c). (35 U.S.C. 154(a)(2) does not take into account, for the determination of the patent term, any application on which priority is claimed under 35 U.S.C. 119, 365(a) or 365(b).) For a CIP application, applicant should review whether any claim in the patent that will issue is supported by an earlier application and, if not, the applicant should consider canceling the reference to the earlier filed application. The term of a patent is not based on a claim-by-claim approach. See Notice of April 14, 1995, 60 Fed Reg. 20,195, at 20,205.

(complete the following, if applicable)

Amend the specification by inserting, before the first line, the following sentence:

A. 35 U.S.C. 120, 121 and 365(c)

() "This is a
- ( ) continuation
- ( ) continuation-in-part
- ( ) divisional

of copending application(s) serial number filed on ."

( ) International Application________ filed on_______ and which designated the U.S."

Note: The proper reference to a prior filed PCT application that entered the U.S. national phase is the U.S. serial number and the filing date of the PCT application that designated the U.S. Moreover, (1) Where the application being transmitted adds subject matter to the International Application, then the filing can be as a continuation-in-part or (2) if it is desired to do so for other reasons then the filing can be as a continuation.

( ) "The nonprovisional application designated above, namely application no.______________, filed___________, claims the benefit of U.S. Provisional Application(s) No(s).:

{list application no(s). and filing date(s)}

B. 35 U.S.C. 119(e) (Provisional Application)

( ) "This application claims the benefit of U.S. Provisional Application(s) No(s).:

{list application no(s). and filing date(s)}

2.2 Relate Back—35 U.S.C. 119 Priority Claim for Prior Application

The prior U.S. application(s), including any prior International Application designating the U.S., identified above in item 2.1(A), in turn itself claim(s) foreign priority(ies) as follows:

{list country, application no(s). and filing date(s)}

The certified copy(ies) has (have)

( ) been filed on_____, in prior application serial no.____, which was filed on___.

( ) is (are) attached.

2.3 Maintenance of Copendency of Prior Application

NOTE: The PTO finds it useful if a copy of the petition filed in the prior application extending the term for response is filed with the papers constituting the filing of the continuation application. Notice of November 5, 1985 (1060 O.G. 27).

A. ( ) Extension of time in prior application

*(This item **must** be completed and the papers filed **in the prior application** if the period set in the prior application has run.)*

( ) A petition, fee and response extends the term in the pending **prior** application until Extension of_____.

( ) A **copy** of the petition filed in prior application is attached.

B. ( ) Conditional Petition for Extension of Time in Prior Application

(complete this item, if previous item not applicable)

( ) A conditional petition for extension of time is being filed in the pending **prior** application.

( ) A **copy** of the conditional petition filed in the prior application is attached.

2.4 Further Inventorship Statement Where Benefit of Prior Application(s) Claimed

(complete applicable item A, B and/or C below)

A. ( ) This application discloses and claims only subject matter disclosed in the prior application whose particulars are set out above and the inventor(s) In this application are

( ) the same.

( ) less than those named in the prior application. It is requested that the following inventor(s) identified for the prior application be deleted:

{ type name(s) of inventor(s) to be deleted }

B. ( ) This application discloses and claims additional disclosure by amendment and a new declaration or oath is being filed. With respect to the prior application, the inventor(s) in this application are

( ) the same.

( ) the following additional inventor(s) have been added:

(type name(s) of inventor(s) to be added)

C. ( ) The inventorship for all the claims in this application are

( ) the same.

( ) not the same. An explanation, including the ownership of the various claims at the time the last claimed invention was made

( ) is submitted.

( ) will be submitted.

2.5 Abandonment of Prior Application *(if applicable)*

( ) Please abandon the prior application at a time while the prior application is pending, or when the petition for extension of time or to revive In that application is granted, and when this application is granted a filing date, so to make this application copending with said prior application.

NOTE: *According to the Notice of May 13, 1983 (103, TMOG 6-7), the filing of a continuation or continuation-in-part application is a proper response with respect to a petition for extension of time or a petition to revive and should include the express abandonment of the prior application conditioned upon the granting of the petition and the granting of a filing date to the continuing application.*

2.6 Petition for Suspension of Prosecution for the Time Necessary to File an Amendment

NOTE: *Where it is possible that the claims on file will give rise to a first action final for this continuation application and for some reason an amendment cannot be filed promptly (e.g, experimental data is being gathered) it may be desirable to file a petition for suspension of prosecution for the time necessary.*

(check the next Item, if applicable)

( ) There is provided herewith a Petition To Suspend Prosecution for the Time Necessary to File An Amendment (New Application Filed Concurrently)

2.7 Small Entity (37 CFR § 1.28(a))

( ) Applicant has established small entity status by the previous submission of a statement in prior application serial no.____ on ____.

( ) A copy of the statement previously filed is included.

WARNING: See 37 CFR § 1.28(a).

2.8. Notification in Parent Application of this Filing

( ) A notification of the filing of this *(check one of the following)*

- ( ) continuation
- ( ) continuation-in-part
- ( ) divisional

is being filed in the parent application, from which this application claims priority under 35 U.S.C. § 120.

2.9 Incorporation by Reference

() the entire disclosure of the prior application, from which a copy of the oath or declaration is supplied, is considered to be part of the disclosure of the accompanying application and is hereby incorporated by reference therein.

3. Papers Enclosed Which are Required for Filing Date Under 37 CFR 1.53(b) (Regular) or 37 CFR 1.153 (Design) Application

(X) _13_ Pages of specification
(X) _4_ Pages of claims
(X) _1_ Pages of Abstract
(X) _10_ Sheets of drawing
- (X) formal
- () informal

4. Additional papers enclosed

() Amendment to claims:

( ) **Cancel** in this application claims_____ before calculating the filing fee. (At least one original independent claim must be retained for filing purposes).

( ) **Add** the claims shown in the attached amendment. (Claims added have been numbered consecutively following the highest numbered original claims).

( ) Preliminary Amendment
(X) Information Disclosure Statement (37 C.F.R. 1.98)
(X) Form PTO-1449
( ) Citations
( ) Declaration of Biological Deposit
( ) Special Comments
( ) Other

5. Declaration or oath (including power of attorney)

(X) ENCLOSED.
() Newly executed (original or copy)
() Copy from prior application No. 0 /______ (37 CFR 1.63(d)- continuation/divisional)

( ) DELETION OF INVENTOR(S) - signed statement attached deleting inventor(s) named in the above-noted prior application (37 CFR 1.63(d) and 1.33(b))

Declaration or Oath executed by: (check **all** applicable boxes)
() inventor(s).
( ) legal representative of inventor(s). 37 CFR 1.42 or 1.43
( ) joint inventor or person showing a proprietary interest on behalf of inventor who refused to sign or cannot be reached.
( ) this is the petition required by 37 CFR 1.47 and the statement required by 37 CFR 1.47 is also attached. See item 13 below for fee.

() NOT ENCLOSED.
( ) Application is made by a person authorized under 37 CFR 1.41(c) on behalf of all the above named inventor(s). The declaration or oath, along with the surcharge required by 37 CFR 1.16(e) can be filed subsequently.
( ) Showing that the filing is authorized. (Not required unless called into question. 37 CFR 1.41(d)).

6. Inventorship Statement

WARNING: If the named inventors are each not the inventors of all the claims an explanation, including the ownership of the various claims at the time the last claimed invention was made, should be submitted.

The inventorship for all the claims in this application are:
() The same
or
() Not the same. An explanation, including the ownership of the various claims at the time the last

( ) **Add** the claims shown in the attached amendment. (Claims added have been numbered consecutively following the highest numbered original claims).

( ) Preliminary Amendment
(X) Information Disclosure Statement (37 C.F.R. 1.98)
(X) Form PTO-1449
( ) Citations
( ) Declaration of Biological Deposit
( ) Special Comments
( ) Other

5. Declaration or oath (including power of attorney)

(X) ENCLOSED.
() Newly executed (original or copy)
() Copy from prior application No. 0 /_______ (37 CFR 1.63(d)- continuation/divisional)

( ) DELETION OF INVENTOR(S) - signed statement attached deleting inventor(s) named in the above-noted prior application (37 CFR 1.63(d) and 1.33(b))

Declaration or Oath executed by: (check **all** applicable boxes)
() inventor(s).
( ) legal representative of inventor(s). 37 CFR 1.42 or 1.43
( ) joint inventor or person showing a proprietary interest on behalf of inventor who refused to sign or cannot be reached.
( ) this is the petition required by 37 CFR 1.47 and the statement required by 37 CFR 1.47 is also attached. See item 13 below for fee.

() NOT ENCLOSED.
( ) Application is made by a person authorized under 37 CFR 1.41(c) on behalf of all the above named inventor(s). The declaration or oath, along with the surcharge required by 37 CFR 1.16(e) can be filed subsequently.
( ) Showing that the filing is authorized. (Not required unless called into question. 37 CFR 1.41(d)).

6. Inventorship Statement

WARNING: If the named inventors are each not the inventors of all the claims an explanation, including the ownership of the various claims at the time the last claimed invention was made, should be submitted.

The inventorship for all the claims in this application are:
() The same
or
() Not the same. An explanation, including the ownership of the various claims at the time the last

claimed invention was made,

() is submitted

() will be submitted.

7. Language

(X) English

() Non-English

() the attached translation is a verified translation. 37 CFR 1.52(d).

8. Assignment

(X) An assignment of the invention

() is attached. (A separate "ASSIGNMENT COVER LETTER ACCOMPANYING NEW PATENT APPLICATION" is also attached.)

(X) will follow.

() The prior application is assigned of record to __(copy attached).

9. Certified Copy - Foreign Priority Claim Under 35 U.S.C. 119

Certified copy(ies) of application(s)

{list country, application no(s). and filing date(s)}

from which priority is claimed

() is (are) attached.

() will follow.

NOTE: The foreign application forming the basis for the claim for priority **must** be referred to in the **oath** or **declaration**. 37 CFR 1.55(a) and 1.63.

NOTE: This item is for any foreign priority for which the application being filed directly relates If any parent U.S. application or International Application form which this application claims benefit under 35 U.S.C. 120 is itself entitled to priority from a prior foreign application then complete item 17 on the ADDED PAGES FOR NEW APPLICATION TRANSMITTAL WHERE BENEFIT OR PRIOR U.S. APPLICATION(S) CLAIMED.

10. Fee Calculation (37 C.F.R. 1.16)

A. (X) Regular Application

CLAIMS AS FILED				
	Number Filed	Number Extra	Rate	Basic Fee $690.00

Total Claims (37 CFR 1.16(c))	21 - 20 =	1	x $ 18.00	$ 18.00
Independent Claims (37 CFR 1.16(b))	3 - 3 =	0	x $ 78.00	$ 0.00
Multiple dependent claim(s), if any (37 CFR 1.16(d))	0	0	x $ 260.00	$ 0.00

( ) Amendment canceling extra claims enclosed.
( ) Amendment deleting multiple dependencies enclosed.
( ) Fee for extra claims is not being paid at this time.

NOTE: If the fees for extra claims are not paid on filing they must be paid or the claims canceled by amendment, prior to the expiration of the time period set for response by the Patent and Trademark Office in any notice of fee deficiency. 37 CFR 1.16(d).

Filing Fee Calculation $ 708.00

B. () Design Application
($330.00 - 37 CFR 1.16(f))

Filing Fee Calculation $

11. Small Entity Statement(s)

() Verified Statement(s) that this is a filing by a small entity under 37 CFR 1.9 and 1.27 is(are) attached.

Filing Fee Calculation (50% of **A** or **B** above) $

NOTE: Any excess of the full fee paid will be refunded if a verified statement and a refund request are filed within **2 months** of the date of timely payment of a full fee. 37 CFR 1.28(a).

12. Request for International-Type Search (37 C.F.R. 1.104(d))

( ) Please prepare an international-type search report for this application at the time when national examination on the merits takes place.

13. Fee Payment Being Made At This Time

( ) NOT ENCLOSED.
( ) No filing fee is to be paid at this time. (This and the surcharge required by 37 CFR 1.16(e) can be paid subsequently.)

(X) ENCLOSED

(X) Filing fee $ 708.00

() Recording assignment
($40.00; 37 CFR 1.21(h)(1)) $

( ) petition fee for filing by other than all the inventors or person on behalf of the inventor where inventor refused to sign or cannot be reached. ($130.00; 37 CFR 1.47 & 1.17(h))
$____

( ) for processing an application with a specification is a non-English language. ($130.00 37 CFR 1.52(d) and 1.17(k))
$____

( ) processing and retention fee. ($130.00; 37 CFR 1.53(d) and 1.21(l))
$____

( ) Fee for international-type search report. ($40.00; 37 CFR 1.21(e))
$____

Total fees enclosed $ 708.00

14. Method of Payment of Fees

(X) Check in the amount of $ 708.00

() Charge Account No. 50-0902 in the amount of $ A duplicate of this transmittal is attached.

15. Authorization to Charge Additional Fees

WARNING: If no fees are to be paid on filing the following items should **not** be completed.

WARNING: Accurately count claims, especially multiple dependent claims, to avoid unexpected high charges, if extra claim charges are authorized.

(X) The Commissioner is hereby authorized to charge the following additional fees by this paper and during the entire pendency of this application to Account No. 50-0902, **identifying our Attorney Docket No.** (74218/05085).

(X) 37 CFR 1.16(a), (f), or (g) (filing fees)
(X) 37 CFR 1.16(b), (c) and (d) (presentation of extra claims)
(X) 37 CFR 1.17 (application processing fees)
( ) 37 CFR 1.16(e) (surcharge for filing the basic filing fee and/or declaration on a date later than the filing date of the application)
( ) 37 CFR 1.17(a)(1)-(5) (extension fees pursuant to 37 CFR 1.136(a))

( ) 37 CFR 1.18 (issue fee at or before mailing Notice of Allowance, pursuant to 37 CFR 1.311(b))

16. Instruction As To Overpayment

() Credit Account No. 50-0902, **identifying our Attorney Docket No. ______.**
(X) Refund

17. Incorporation by reference of added pages

() The following pages are incorporated by reference:

() "Assignment Cover Letter Accompanying New Application"; number of pages added

(X) Added Pages For Papers Referred To In Item 4 Above; number of pages added 4

( ) Plus added pages deleting names of inventor(s) named in prior application(s) who is/are no longer inventor(s) of the subject matter claimed in this application; number of pages added_ ______.

(X) no further pages form a part of this Transmittal. The transmittal ends with this page.

Date: July 31, 2000

[illegible]
Michael A. Jaffe
Registration No. 36,326

ARTER & HADDEN LLP
1100 Huntington Building
925 Euclid Avenue
Cleveland, Ohio 44115-1475
Phone: (216) 696-3394
Fax: (216) 696-2645

523119.1

DOCKET NO. 74218/05085

ON-LINE SYSTEM FOR CREATING A PRINTABLE PRODUCT

Field of Invention

The present invention generally relates to a system for creating printable products, such as announcements, banners, business cards, calendars, greeting cards, certificates, craft cards, envelopes, gift tags, invitations, labels, message cards, origami, postcards, posters, stationary, and stickers. More particularly the present invention relates to a system accessible via a computer network for creating customized printable products.

Background of the Invention

Systems for creating printable products are generally comprised of five basic components, namely, (1) a "composition engine" component for composing the printable product, (2) a "menu" component for facilitating operation of the system, (3) an "assets" component which provide the visual and formatting content (e.g., graphic elements, text elements, text and graphics formatting data) for the printable product, (4) an "assembly" component for assembling a printable product file suitable for printing, and (5) a "printing" component for printing the printable product.

Current products for creating printable products are sold as software packages installed by a user on their personal computer. Examples of such products include American Greetings® CreataCard® and Mindscape® Printshop®. In these products, the composition engine, menu, asset, assembly and printing components initially reside one or more computer disks (e.g., floppy disk, CD-ROM, DVD). All or portions of these components are loaded into the hard drive of a personal computer for execution by the CPU.

The foregoing approach to creation of printable products has several drawbacks. In order to provide a user with a very large selection of assets for a variety of different printable products, a plurality of disks are needed. Thus, a user must shuffle several disks in and out of the personal computer disk/CD ROM drive in order to review and select the desired assets for a printable product. Alternatively, a user can load the assets to their hard drive which consumes significant storage resources of the user's personal computer.

Another drawback is that the selection of assets remain static, and thus get "stale" over time. Many users desire new assets for the printable products. Thus, a user must periodically acquire new disks with new assets in order create printable products with "fresh" assets. Similarly, the engine component may be frequently upgraded with enhanced features (e.g., new types of printable products), and thus the user must acquire new disks with the upgraded engine component in order to utilize the enhanced features.

While it is possible to download new assets and engines over a computer network such as the Internet, the downloading process can be very slow, and significant hard disk resources of the user's personal computer are consumed in order to store the downloaded data.

The present invention addresses these and other drawbacks of the prior art by providing an on-line system for creation of printable products.

Summary of the Invention

According to the present invention there is provided a system for on-line creation of a printable product, the system comprising: (1) at least one server accessible via a computer network, said at least one server storing defining data defining a plurality of printable products including one or more design elements, and a first program providing modification functions for modifying the defining data, and assembly functions for assembling a printable product suitable for printing; and (2) a client computer for accessing said server, wherein said at least one server downloads said first program to said client computer.

In accordance with another aspect of the present invention, there is provided a computer usable medium having computer readable program code means embodied therein for creating, modifying and printing of a printable product, the computer readable program code means comprising: (1) means for downloading data defining a printable product from a remote storage device; (2) modification means for modifying the defining data; and (3) print formatting means for formatting the defining data for printing.

In accordance with another aspect of the present invention, there is provided a method for generating a printable product using an on-line system accessible via a computer network, the method including the steps of: (a) storing on a server accessible via the computer network, data defining a plurality of printable products including one or more design elements; (b) storing on the server a first program providing modification functions for modifying the defining data, and assembly functions for assembling a printable product suitable for printing; and (c) downloading the first program to a client computer accessing the server, to allow for modification and printing of a printable product at the client computer.

An advantage of the present invention is the provision of an on-line system for creating a printable product that minimizes the consumption of storage resources of a user's computer.

Another advantage of the present invention is the provision of an on-line system for creating a printable product that provides a user with fast and convenient access to updated printed product assets.

Still another advantage of the present invention is the provision of an on-line system for creating a printable product that provides a user with fast and convenient access to enhanced engines.

Still another advantage of the present invention is the provision of an on-line system for creating a printable product that utilizes the functionality of a browser.

Still another advantage of the present invention is the provision of an on-line system for creating a printable product that extends the functionality of a browser by use of plug-ins.

Yet another advantage of the present invention is the provision of an on-line system for creating a printable product which provides a user's computer with extensive editing functions for editing data defining a printable product, including editing functions for formatting a variety of different design elements.

Still other advantages of the invention will become apparent to those skilled in the art upon a reading and understanding of the following detailed description, accompanying drawings and appended claims.

Brief Description of the Drawings

The invention may take physical form in certain parts and arrangements of parts, a preferred embodiment and method of which will be described in detail in this specification and illustrated in the accompanying drawings which form a part hereof, and wherein:

Fig. 1 shows a flow diagram of the user-initiated steps for creating a printable product, according to a preferred embodiment of the present invention;

Fig. 2 shows a generally overview of a system arrangement, according to a preferred embodiment of the present invention;

Fig. 3 illustrates an exemplary record for a text element, in accordance with a preferred embodiment;

Fig. 4 illustrates an exemplary record for a graphic element, in accordance with a preferred embodiment;

Fig. 5A illustrates an exemplary menu of card selections for customization and printing;

Fig. 5B illustrates an exemplary menu of birthday card selections for customization and printing;

Fig. 5C illustrates basic properties of a selected birthday card;

Fig. 5D illustrates a web page which displays the assets of a selected birthday card, and provides means for user modification of the assets.

Fig. 6A illustrates the outside panels (front and rear) of a single fold card;

Fig. 6B illustrates the inside panels of a single (half) fold card; and

Fig. 7 illustrates all four panels of a double (quarter) fold card.

Detailed Description of the Preferred Embodiment

It should be appreciated that while a preferred embodiment of the present invention will be described in connection with the creation of a printed product taking the form of a greeting card, the printed product make take other forms, including but not limited to: announcements, banners, business cards, calendars, certificates, craft cards, envelopes, gift tags, invitations, labels, message cards, origami, postcards, posters, stationary, stickers and other social expression products. The printed product includes one or more design elements, including but not limited to: text, graphic/image, audio and video. The use of greeting cards in describing a preferred embodiment is not meant in any way to limit the scope of the present invention.

Moreover, it should be appreciated that while a preferred embodiment of the present invention has been described in connection with the Internet, the present invention may be used in conjunction with other computer networks, including other public computer networks, as well as proprietary/private computer networks.

As known in the prior art, the World Wide Web (WWW) comprises many pages or files of information, distributed across many different server computer systems.

A wide variety of different types of information may be stored on such pages, and this information can be presented to a user's computer system (typically referred to as "client computer system") using a combination of text, graphics, audio data and video data. Each page is identified by a Universal Resource Locator (URL). The URL denotes both the server machine, and the particular file or page on that machine. There may be many pages or URLs resident on a single server. In order to use the WWW, a client computer system runs a piece of software known as a graphical Web browser, such as "Navigator" available from Netscape Communications Corporation, or "Internet Explorer" available from Microsoft Corporation. "Navigator" is a trademark of the Netscape Communications Corporation, while "Internet Explorer" is a trademark of Microsoft Corporation.

The client computer system interacts with the browser to select a particular URL, which in turn causes the browser to send a request for that URL or page to the server identified in the URL. Typically the server responds to the request by retrieving the requested page, and transmitting the data for that page back to the requesting client computer system (the client/server interaction is performed in accordance with the hypertext transport protocol ("HTTP")). This page is then displayed to the user on the client screen. The client may also cause the server to launch an application. Most WWW pages are formatted in accordance with a computer program written in a language known as HTML (hypertext mark-up language). This program contains the data to be displayed via the client's graphical browser as well as formatting commands which tell the browser how to display the data. Thus, a typical Web page includes text together with embedded formatting commands, referred to as tags, which can be used to control the font size, the font style (for example, whether italic or bold), how to lay-out the text, and so on. A Web browser "parses" the HTML script in order to display the text in accordance with the specified format. HTML tags are also used to indicate how graphics, audio and video are manifested to the user via the client's browser.

Most Web pages also contain one or more references to other Web pages, which need not be on the same server as the original page. Such references may generally be activated by the user selecting particular locations on the screen, typically by clicking a mouse control button. These references or locations are known as hyperlinks, and are typically flagged by the browser in a particular manner (for example, any text associated with a hyperlink may be in a different color). If a user selects the hyperlink, then the referenced page is retrieved and replaces the currently displayed page.

A preferred embodiment of the present invention takes advantage of the features of the Internet and Internet web browsers. Moreover, a preferred embodiment of the present invention enhances the utility of the browser by use of a plug-in program, as will be described in detail below. A plug-in program is used to alter, enhance, or extend the operation of a parent application program. The idea behind plug-in's is that a small piece of software is loaded into memory by the larger program, adding a new feature, and that users need only install the few plug-ins that they need, out of a much larger pool of possibilities. Browsers such as Netscape Navigator World-Wide Web browser and Microsoft Internet Explorer supports plug-ins which display or interpret a particular file format or protocol such as Shockwave, RealAudio, Adobe Systems, Inc. PDF, Corel CMX (vector graphics). The file to be displayed is included in a web page using an EMBED HTML tag.

Referring now to the drawings wherein the showings are for the purposes of illustrating a preferred embodiment of the invention only and not for purposes of limiting same, Fig. 1 provides a flow diagram of the user-initiated steps for creating a printable product, according to a preferred embodiment of the present invention. These steps include SELECT CONTENT (step 12) which provides web page displays to the user for selecting genre, selecting a specific card of the selected genre, and displaying the card attributes for the specific selected card. With respect to selection of genre, the user is presented with a listing (and optionally descriptions and samples) of available genres.

These genres may divided into categories. Examples of genre categories are: Holidays, Just Because, Friendship, Love, Birthday, Romantic Events, Baby, To Kids, Life Events, Concern & Support, Collections, Inspirational & Religions, Spanish, and Business (Fig. 5A). Each of the genre categories may be subdivided into one or more subcategories. For instance, the Holidays category may have such subcategories as Graduation, Grandparents Day, Jewish New Year, Boss's Day, Sweetest Day, Halloween, Thanksgiving, Hanukkah, Christmas, Kwanza, Hari Raya, New Year's Day, Chinese New Year, Valentines Day, St. Patricks Day, Passover, Easter, Secretaries Day, Mother's Day and Father's Day. To facilitate selection of the categories and subcategories drop down selection boxes and/or hypertext links are displayed to the user. In addition, information associated with the category/subcategory may also be displayed, such as shown in Fig. 5B (e.g., date of the holiday, samples or popular card selections for the category/subcategory, and thumbnail displays of cards for the category/subcategory). Once a specific card is selected by the user, basic display attributes of the selected card are displayed to the user (Fig. 5C). In a preferred embodiment, these display attributes include a card title, cover verse, inside verse and cover graphics. If the user's computer does not have the plug-in program for creation of the printed product, the plug-in will be installed at step 14. The process will then proceed to step 16 described below.

SELECT ASSETS (step 16) provides web page displays to the user for modifying each panel of the card selected in step 12 (Fig. 5D). In this regard, the user is prompted to select one of the following panels for modification: (1) front, (2) inside top, (3) inside bottom, and (4) back. The graphic elements and text elements appearing on each selected panel is displayed to the user. The user may modify the font, point size, color and alignment (i.e., right, center, left) for the text elements appearing on each panel. In accordance with a preferred embodiment, the user highlights the displayed text to be modified, and then selects the desired font, point size, color and alignment from a menu of drop down selections.

Once the assets have been selected, and the user requests printing, the user is prompted to enter print parameters, namely, single or quarter fold, and the number of copies. The first time the user specifies single fold, the process proceeds to step 18 described below. This is necessary to properly orient the graphic and text elements for a particular printer. CARD PRINT (step 20) prints the single or quarter fold card at the user's local printer. For a single fold print job, the outside of the card is printed first, and the user reinserts the printed page into the local printer as directed, to print the inside of the card. TEST PRINT (step 18) has the user print a first and second test page using their local printer, so that the proper orientation for a single fold print job can be determined.

Fig. 6A illustrates the outside panels (front and rear) of a single fold card; Fig. 6B illustrates the inside panels of a single (half) fold card; and Fig. 7 illustrates all four panels of a double (quarter) fold card.

Referring now to Fig. 2 there is shown a generally overview of a system arrangement, according to a preferred embodiment of the present invention. As is well known to those skilled in the art, a personal computer (PC) 40 may communicate with a web server 70 via a computer network (i.e., Internet) 60. A plurality of different types of data may be stored on the web server, including but not limited to, plug-ins 50, "CPT" files, thumbnails (which provide a preview of available printable products in one or more sizes), design elements (e.g., graphics and text elements), advertising, promotional and logo data, and bar code data, as will be described in further detail below.

An appropriate plug-in 50 is downloaded to PC 40 to enhance the functionality of browser 42. In this regard, plug-in 50 includes an engine and assembly component for creating the printed product (including local printing). One function of the engine component is to make selected assets (i.e., design elements) for a printed product available in the browser such that they can be edited by the user. Such assets may include, but are not limited to graphic/image elements, text elements, audio elements, and video elements. Editing functions controlled by the plug-in include but are not limited to, modifying text fonts, modifying text point size, modifying text/graphics color, modifying text/graphics alignment, modifying text/graphics position within a panel, adding new text/graphics elements, deleting text/graphic elements. The editing functions may also include those typically found in word processing and design application software. Additional editing functions suitable to other types of design elements may also be provided.

As mentioned above, plug-in 50 is downloaded to the user' PC 40. As is well known to those skilled in the art, browser 42 includes a table of plug-ins that are invoked upon specified conditions, and thus extend the functionality of the browser. For instance, after plug-in 50 has been downloaded to PC 40, if browser 42 detects a file with a .cpt extension, the plug-in related to that file type will be invoked. In a preferred embodiment, plug-in 50 includes a plurality of ActiveX controls that allow browser 42 to interpret CPT files, which are downloaded as a compressed binary file. It should be understood that the extension "cpt" is selected solely to illustrate a preferred embodiment of the present invention, and that any suitable identifier could be used to invoke the plug-in.

In accordance with a preferred embodiment of the present invention, plug-in 50 is downloaded on demand when it is recognized that the browser accessing web server 70 does not already have plug-in 50 installed, or does not have the latest version of plug-in 50 installed. This auto-detect feature makes it simple for users to have the most up-to-date plug-in with the most current enhancements. It should be appreciated that various types of compression algorithms may be utilized to speed up the downloading process.

It should be understood that in accordance with a preferred embodiment, plug-in 50 (in conjunction with browser 42) will include all, or a portion of, the engine component and assembly component. Preferably, the menu and assets components will reside on web server 70.

Referring now to Figs 3 and 4, exemplary database records are shown. Fig. 3 illustrates an exemplary record for a text element, while Fig. 4 illustrates an exemplary record for a graphic element. In this regard, a text element record includes a record identifier, a category (e.g., birthday card, invitation, calendar, origami, etc.), panel no. (e.g., for greeting cards there are four panels; two inside the card and two outside the card), font, point size, color (e.g., hex - RGB), position information (i.e., identify location within the panel relative to one or more reference locations), alignment of the text (i.e., left, right and center), and the text string. Likewise, the graphic element record includes a panel number, position information, and filename of the graphic. It should be understood that the foregoing records may include additional fields, including but not limited to print format data (e.g., specifying whether the text or graphic element is suitable for printing in half-fold and/or quarter fold configurations). This print format data could specify any restrictions or options relating to formatting at the time of printing. Alternatively, the print format data could be externalized and stored separately on the web server.

Web server 70 also may include additional databases for advertising, promotional, logo and/or bar code data that is also printed on the printed product. The advertising, promotional, logo and/or bar code data may be used in connection with several types of printed products. Such records may include an ID field and the display information that is to be printed. Additional fields may be provided for each record to identify the type of printed products and/or the category of printed products. Furthermore, records could be provided which specify that external data (i.e., data outside the cpt file) is to be added to the printed product from another source. For example, the external source data could be an uploaded photograph (e.g., fax gif file) or other graphics file that is added to a birthday card. Another example of external source data is a signature graphic. Reference is made to Figs. 6A and 7, which illustrate examples of advertising, promotional and logo data that is part of the printed product.

Web server 70 pre-assembles the design elements for a printable product selected by the user. In this regard, an appropriate CPT file is generated and downloaded to PC 40 using plug-in 50. The data in the CPT file is compiled from the design element databases. The CPT file includes all the information for the assets of a printable product (e.g., asset information for all four panels of a greeting card), including the information to display the printable product to the user, allow editing of the assets, and assembly for printing. Importantly, the display, editing and assembly of the printable product defined by the CPT file is performed by plug-in 50. It should be noted that the assembly process performed by plug-in 50 includes (but is not limited to) scaling, resizing and division into panels to accommodate the selected paper fold format (e.g., quarter fold, or half fold).

It should be appreciated that in accordance with an alternative embodiment of the present invention, selected "raw" data (e.g., design elements, advertising, promotional, logos, and bar code data) stored in web server 70 which is used to define a printable product may be downloaded directly to plug-in 50. In this regard, no preassembly takes place to form a CPT file. Since all assembly of the design elements occurs at the user's PC, this approach will be slower than the preferred embodiment.

The present invention can be modified to include many additional enhancements. For instance, the user can be provided with powerful editing tools to manipulate the graphical and text elements of the printed product. In this regard, a user may add, delete or reposition design elements. Another enhancement of the present invention is to allow the user to store the data defining the completed printed product in a data file. This data file could then be stored in local storage at the user's PC, stored in a portable storage medium (e.g., floppy disk or CD ROM), or stored at a remote location (e.g., at the web server). Furthermore, this data file may be attached to an email for transmission to another party. This portability of the data defining the printed product allows for delayed printing and printing at a different location. This also facilitates the use of portable (including wireless) web appliances, such as personal digital assistants

(PDAs), such as the Palm Pilot, and web phones, to create a printed product at a location where there is no printer available, and to print the printable product at a different location where a printer is available.

It should be further understood that the advertising, promotional, logo and/or bar code data could be added to the data defining a completed printed product that is stored and/or transmitted to another location. The bar code could be added prior to storing/transmitting the data, or could be added by another plug-in at the remote receiver's computer. In this regard, the bar code could be used to identify shipping information (e.g., data for matching a product to be delivered with the printed product, such as an order of flowers that are to be delivered with a greeting card). Graphics and text elements identifying the shipper's name could also be provided. Therefore, the present invention also finds utility in area of remote-remote fulfillment.

The attached Appendix includes program code listings associated with the preferred embodiment of the "plug-in" described above.

The invention has been described with reference to a preferred embodiment. Obviously, modifications and alterations will occur to others upon a reading and understanding of this specification. It is intended that all such modifications and alterations be included insofar as they come within the scope of the appended claims or the equivalents thereof.

Having thus described the invention, it is now claimed:

1. A system for on-line creation of a printable product, the system comprising:

at least one server accessible via a computer network, said at least one server storing defining data defining a plurality of printable products including one or more design elements, and a first program providing modification functions for modifying the defining data, and assembly functions for assembling a printable product suitable for printing; and

a client computer for accessing said server, wherein said at least one server downloads said first program to said client computer.

2. A system according to claim 1, wherein said plurality of printable products includes at least one of: announcements, banners, business cards, calendars, greeting cards, certificates, craft cards, envelopes, gift tags, invitations, labels, message cards, origami, postcards, posters, stationary, and stickers.

3. A system according to claim 1, wherein said client computer includes a browser program for accessing said web server, wherein said first program enhances the functionality of said browser program.

4. A system according to claim 3, wherein said first program controls the downloading to the client computer of the defining data that defines a selected printable product.

5. A system according to claim 1, wherein said defining data defines at least one of: graphical elements, text elements, and formatting data associated with the graphical and text elements.

6. A system according to claim 1, wherein said system further comprises a printer associated with said client computer.

7. A system according to claim 6, wherein said first program assembles printing data for printing the printable product on the printer.

8. A system according to claim 7, wherein said assembly of printing data includes at least one of: resizing, scaling, division into panels that anticipate printing in a desired printing format.

9. A system according to claim 1, wherein said modification function of said first program includes modification to at least one of: font, color, alignment, position within a panel, adding a design element, and deleting a design element.

10. A computer usable medium having computer readable program code means embodied therein for creating, modifying and printing of a printable product, the computer readable program code means comprising:

means for downloading data defining a printable product from a remote storage device;

modification means for modifying the defining data; and

print formatting means for formatting the defining data for printing.

11. A computer readable program code means according to claim 10, wherein said modification means includes means for manipulating one or more design elements.

12. A computer readable program code means according to claim 11, wherein said design elements includes at least one of: text, graphics, audio and video.

13. A computer readable program code means according to claim 10, wherein said print formatting means performs at least of the following functions: resizing, scaling, and division into panels associated with a fold format.

14. A method for generating a printable product using an on-line system accessible via a computer network, the method comprising:

storing on a server accessible via the computer network, data defining a plurality of printable products including one or more design elements;

storing on the server a first program providing modification functions for modifying the defining data, and assembly functions for assembling a printable product suitable for printing; and

downloading the first program to a client computer accessing the server, to allow for modification and printing of a printable product at the client computer.

15. A method according to claim 14, wherein said plurality of printable products includes at least one of: announcements, banners, business cards, calendars, greeting cards, certificates, craft cards, envelopes, gift tags, invitations, labels, message cards, origami, postcards, posters, stationary, and stickers.

16. A method according to claim 14, wherein said client computer includes a browser program for accessing said web server, wherein said first program enhances the functionality of said browser program.

17. A method according to claim 16, wherein said first program controls the downloading to the client computer of the defining data that defines a selected printable product.

18. A method according to claim 14, wherein said defining data defines at least one of: graphical elements, text elements, and formatting data associated with the graphical and text elements.

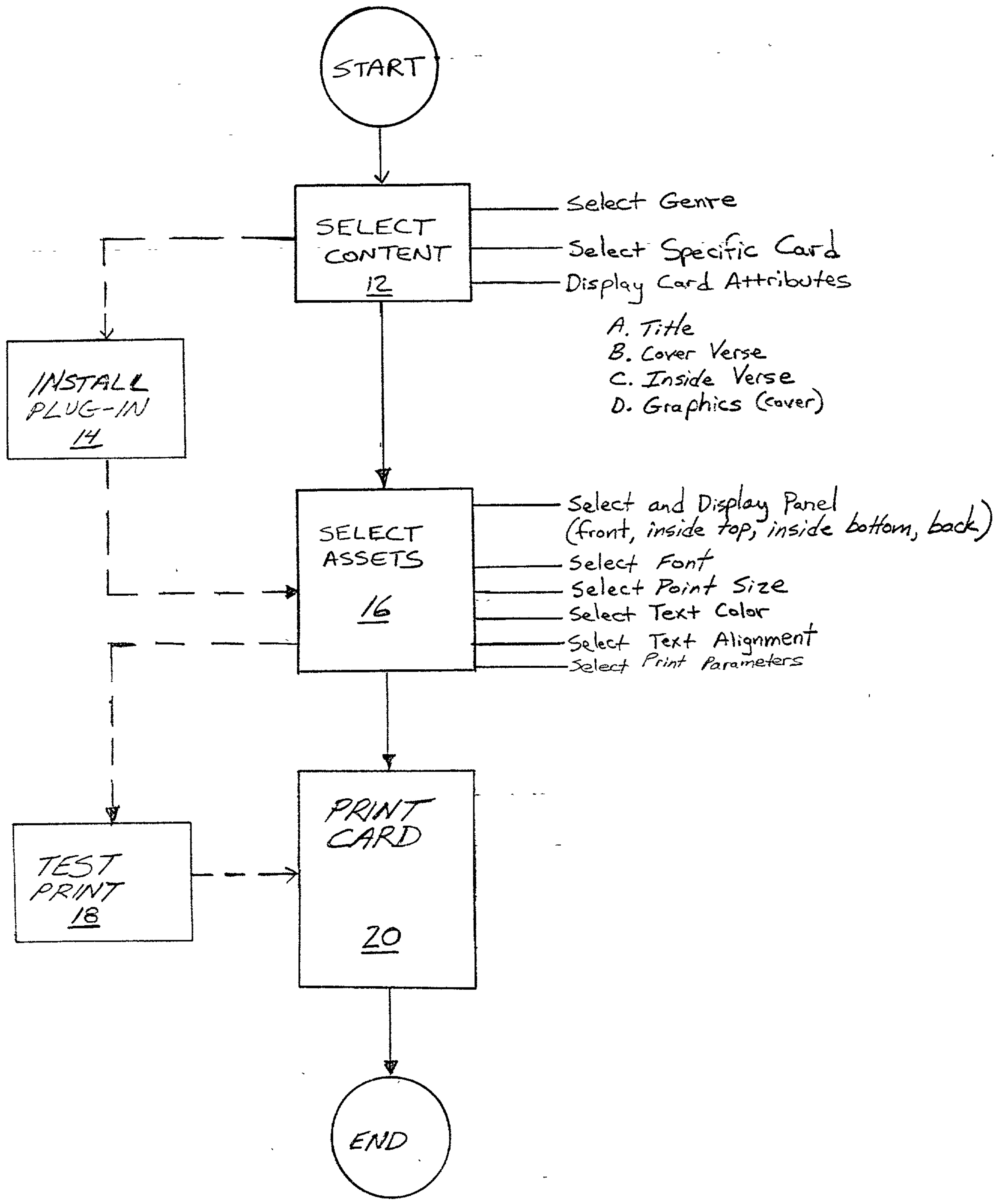
19. A method according to claim 14, wherein said method further comprises using the first program to assemble printing data for printing the printable product on the printer.

20. A method according to claim 19, wherein said step of assembling printing data includes at least one of: resizing, scaling, division into panels that anticipate printing in a desired printing format.

21. A method according to claim 14, wherein said modification function of said first program includes modification to at least one of: font, color, alignment, position within a panel, adding a design element, and deleting a design element.

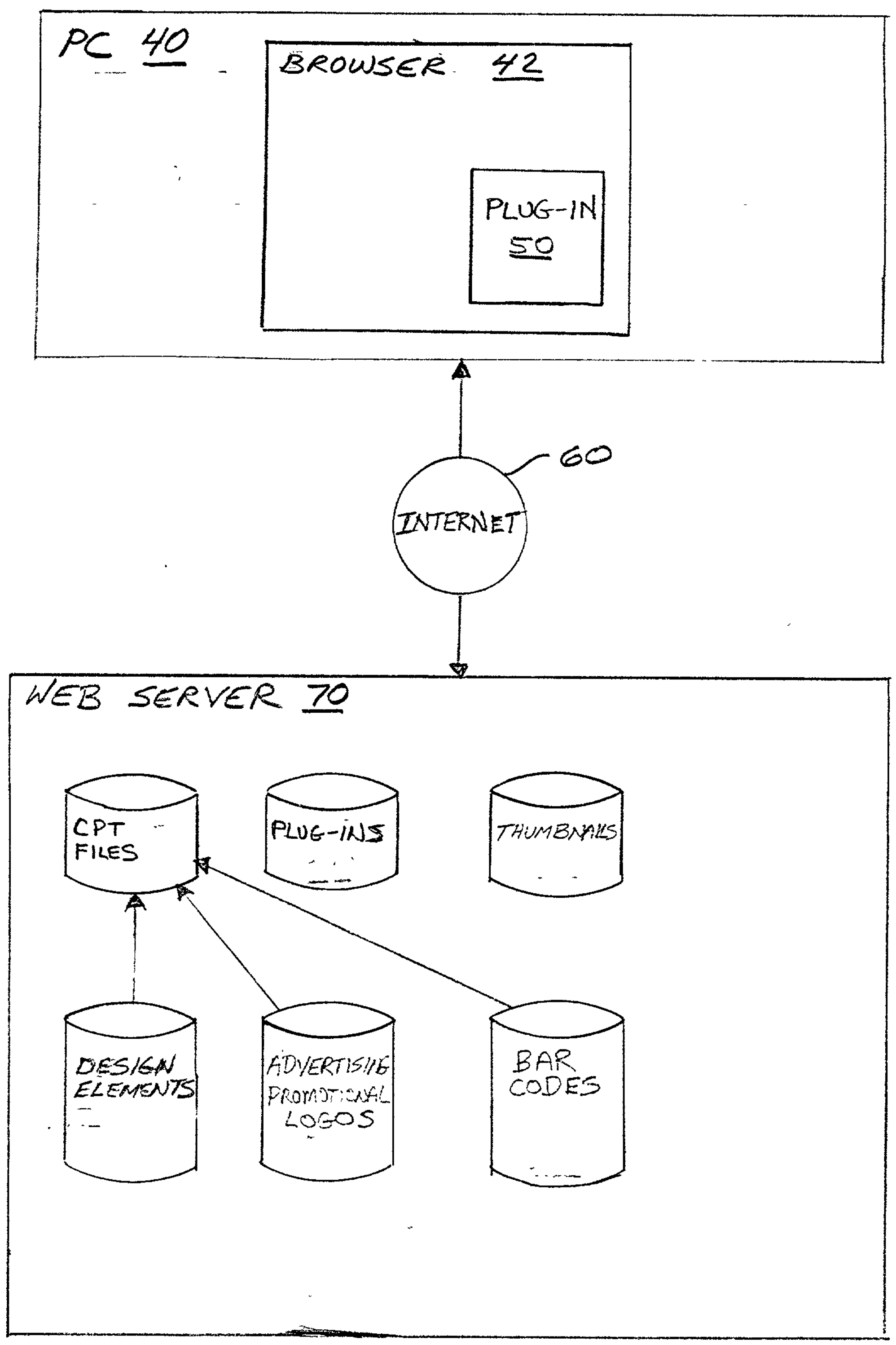
ABSTRACT

A system for providing on-line creation of printable products such as, announcements, banners, business cards, calendars, greeting cards, certificates, craft cards, envelopes, gift tags, invitations, labels, message cards, origami, postcards, posters, stationary, stickers, and other social expression products. The printable products are selectable from a list, and modifiable on-line to provide a user-customized product. The customized printable product may be printed at the user's local printer or stored in a file for later access. Modifications to the printable product include modification of text elements (e.g., formatting, such as font, point size, color, and alignment), and graphical elements. A plug-in is downloaded to a user's PC to enhance the functionality of the user's browser to customize and print the printable product.



START
SELECT CONTENT 12
Select Genre
Select Specific Card
Display Card Attributes
A. Title
B. Cover Verse
C. Inside Verse
D. Graphics (cover)
INSTALL PLUG-IN 14
SELECT ASSETS 16
Select and Display Panel (front, inside top, inside bottom, back)
Select Font
Select Point Size
Select Text Color
Select Text Alignment
Select Print Parameters
TEST PRINT 18
PRINT CARD 20
END

Fig. 1



PC 40
BROWSER 42
PLUG-IN 50
60
INTERNET
WEB SERVER 70
CPT FILES
PLUG-INS
THUMBNAILS
DESIGN ELEMENTS
ADVERTISING PROMOTIONAL LOGOS
BAR CODES

Fig. 2

TEXT ELEMENT #1

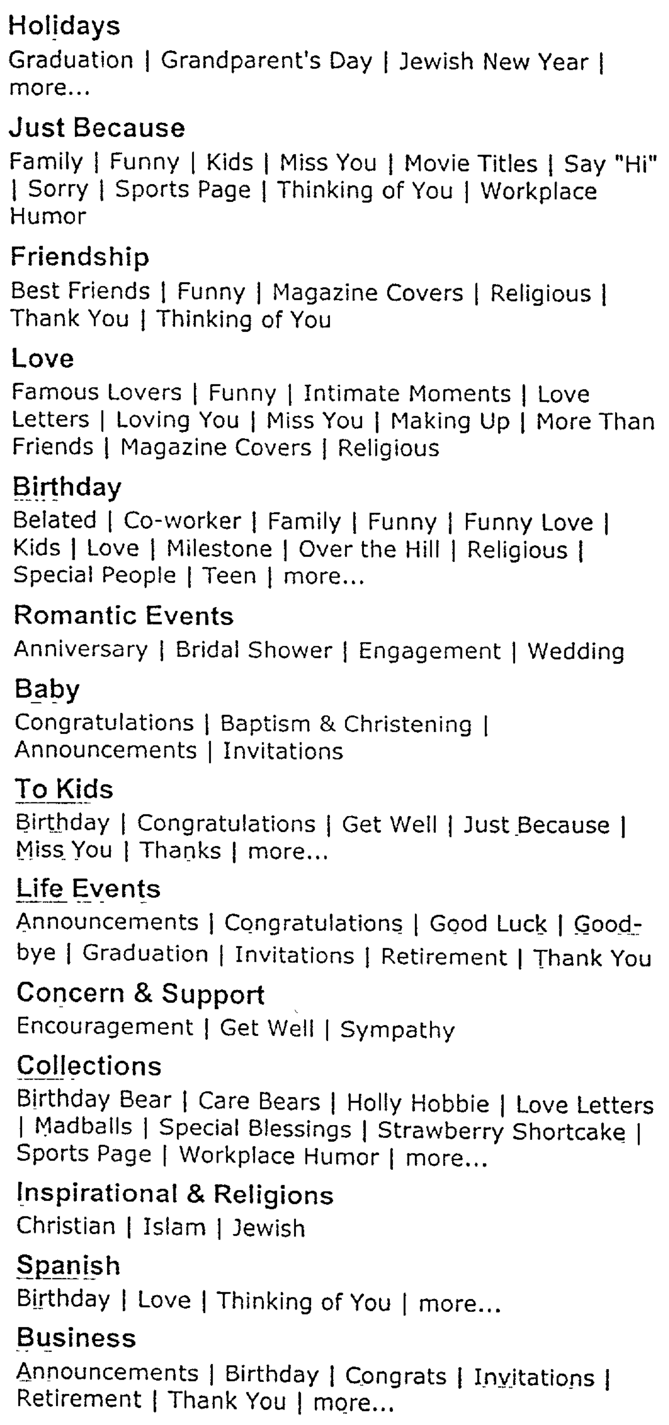
ID	CATEGORY	PANEL NO.	FONT	PT. SIZE	COLOR	STARTING POSITION UPPER LEFT OVER	STARTING POSITION UPPER LEFT DOWN	STARTING POSITION UPPER RIGHT OVER	STARTING POSITION UPPER RIGHT DOWN	ALIGN	TEXT STRING
BDAY1	B-DAYCARD	1	ARIAL	10	FFFFFF	.5	.5	.4	.5	L	HAPPY BIRTHDAY BRUCE

Fig. 3

GRAPHIC ELEMENT #1

ID	CATEGORY	PANEL NO.	UPPER LEFT POSITION OVER	UPPER LEFT POSITION DOWN	LOWER RIGHT POSITION OVER	LOWER RIGHT POSITION DOWN	FILENAME
		1	2	3	4	5	MAN.GIF

Fig. 4



Holidays
Graduation | Grandparent's Day | Jewish New Year | more...

Just Because
Family | Funny | Kids | Miss You | Movie Titles | Say "Hi" | Sorry | Sports Page | Thinking of You | Workplace Humor

Friendship
Best Friends | Funny | Magazine Covers | Religious | Thank You | Thinking of You

Love
Famous Lovers | Funny | Intimate Moments | Love Letters | Loving You | Miss You | Making Up | More Than Friends | Magazine Covers | Religious

Birthday
Belated | Co-worker | Family | Funny | Funny Love | Kids | Love | Milestone | Over the Hill | Religious | Special People | Teen | more...

Romantic Events
Anniversary | Bridal Shower | Engagement | Wedding

Baby
Congratulations | Baptism & Christening | Announcements | Invitations

To Kids
Birthday | Congratulations | Get Well | Just Because | Miss You | Thanks | more...

Life Events
Announcements | Congratulations | Good Luck | Good-bye | Graduation | Invitations | Retirement | Thank You

Concern & Support
Encouragement | Get Well | Sympathy

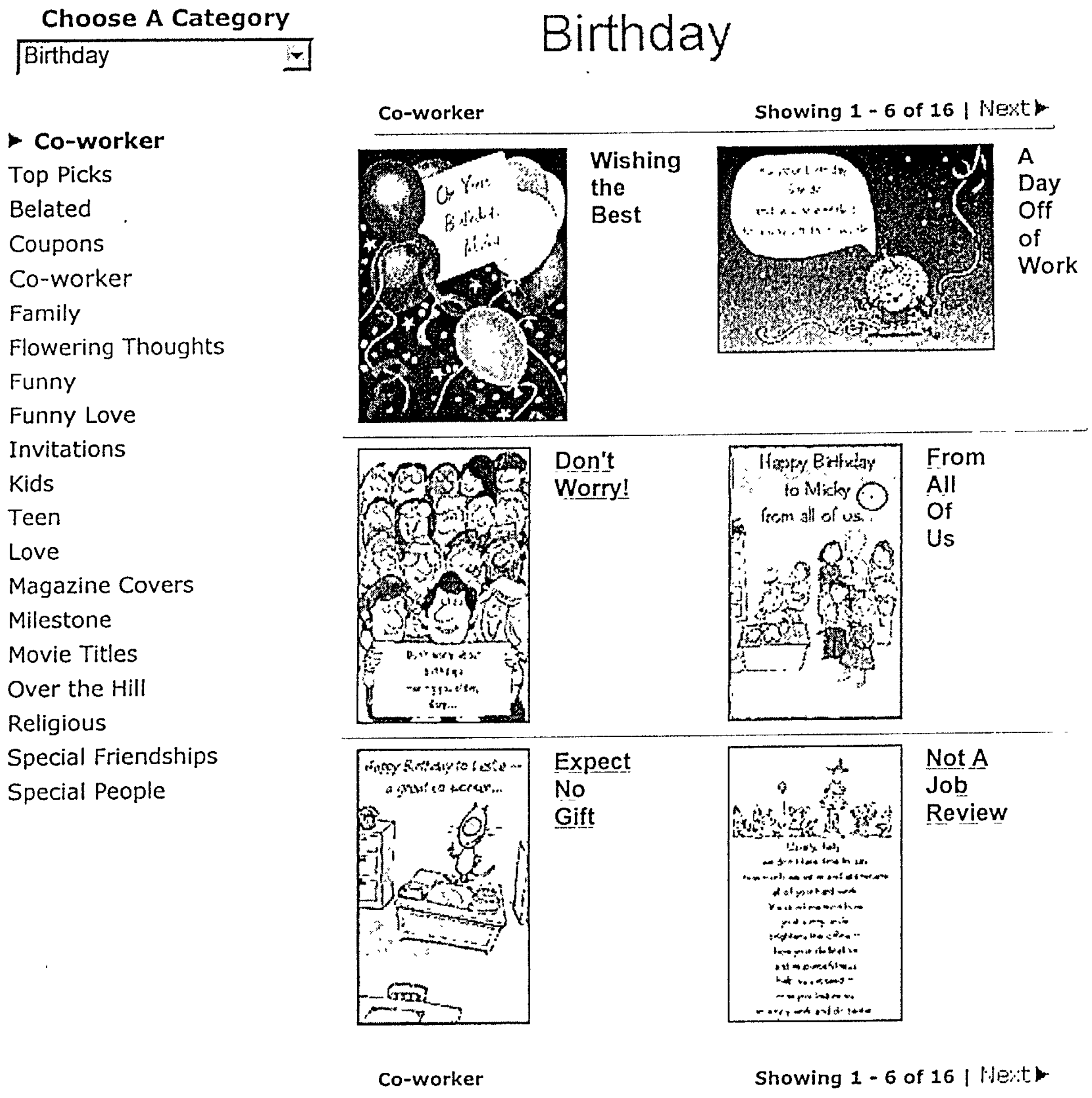
Collections
Birthday Bear | Care Bears | Holly Hobbie | Love Letters | Madballs | Special Blessings | Strawberry Shortcake | Sports Page | Workplace Humor | more...

Inspirational & Religions
Christian | Islam | Jewish

Spanish
Birthday | Love | Thinking of You | more...

Business
Announcements | Birthday | Congrats | Invitations | Retirement | Thank You | more...

Fig. 5A



Choose A Category

Birthday

▶ Co-worker
Top Picks
Belated
Coupons
Co-worker
Family
Flowering Thoughts
Funny
Funny Love
Invitations
Kids
Teen
Love
Magazine Covers
Milestone
Movie Titles
Over the Hill
Religious
Special Friendships
Special People

Birthday

Co-worker — Showing 1 - 6 of 16 | Next ▶

	Wishing the Best		A Day Off of Work
	Don't Worry!		From All Of Us
	Expect No Gift		Not A Job Review

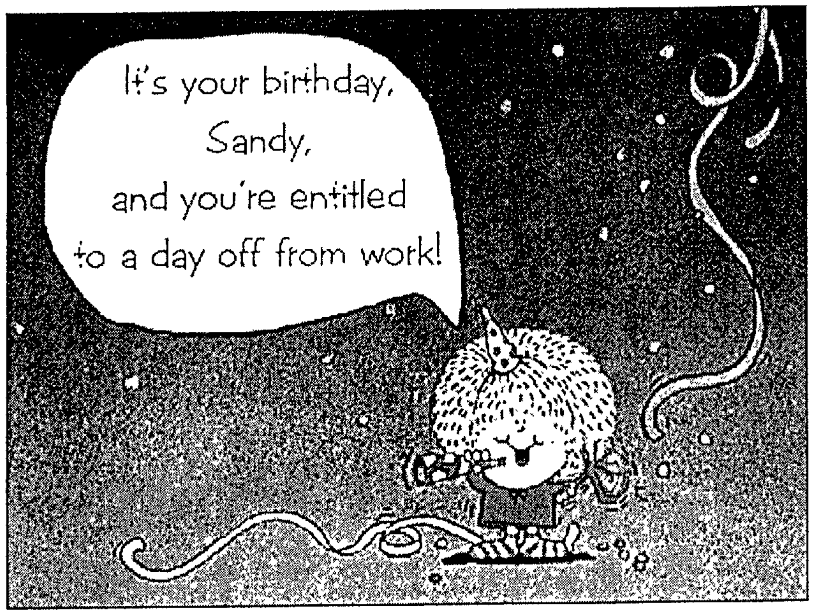
Co-worker — Showing 1 - 6 of 16 | Next ▶

Fig. 5B

Create and Print this project!

Enjoy Create & Print cards! Personalize, print, & send as many cards as you'd like.
It's fast, easy, and fun!

previous | more greetings | next | personalize your greeting



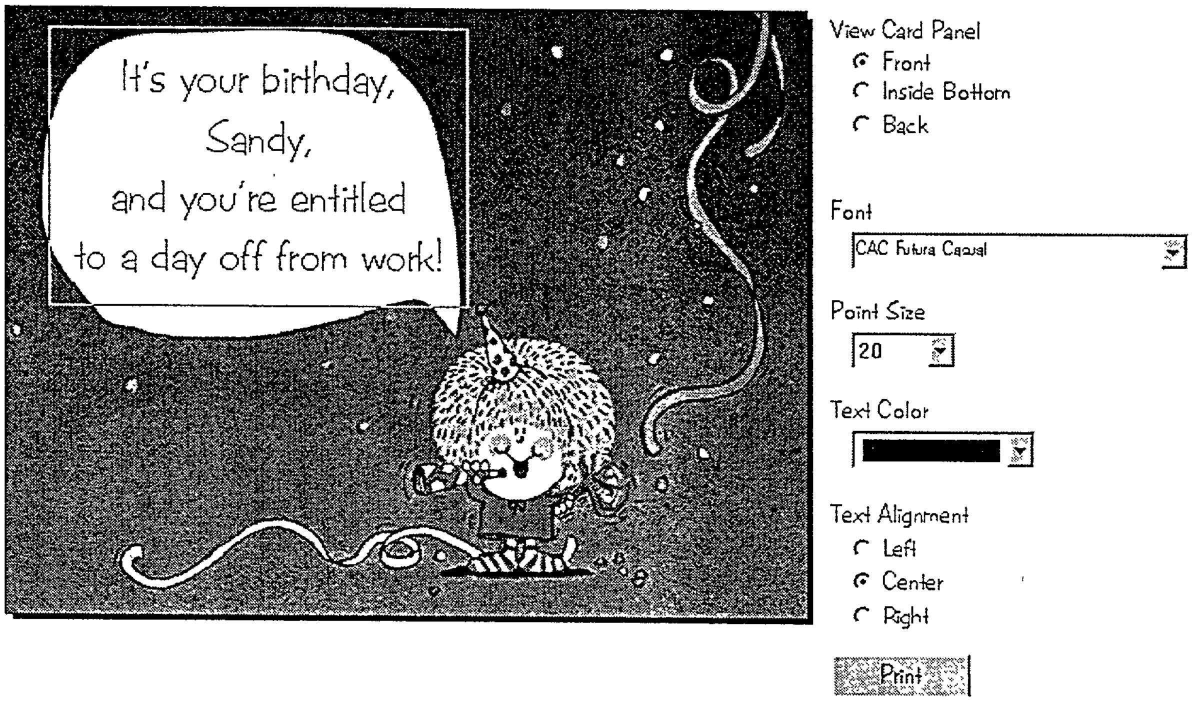
"A Day Off of Work"

Cover Verse: It's your birthday, Sandy, and you're entitled to a day off from work!

Inside Verse:

You're not getting it, of course, but you're entitled to it.

Fig. 5C



It's your birthday,
Sandy,
and you're entitled
to a day off from work!

View Card Panel
[✓ Front]
Inside Bottom
Back

Font
CAC Futura Casual

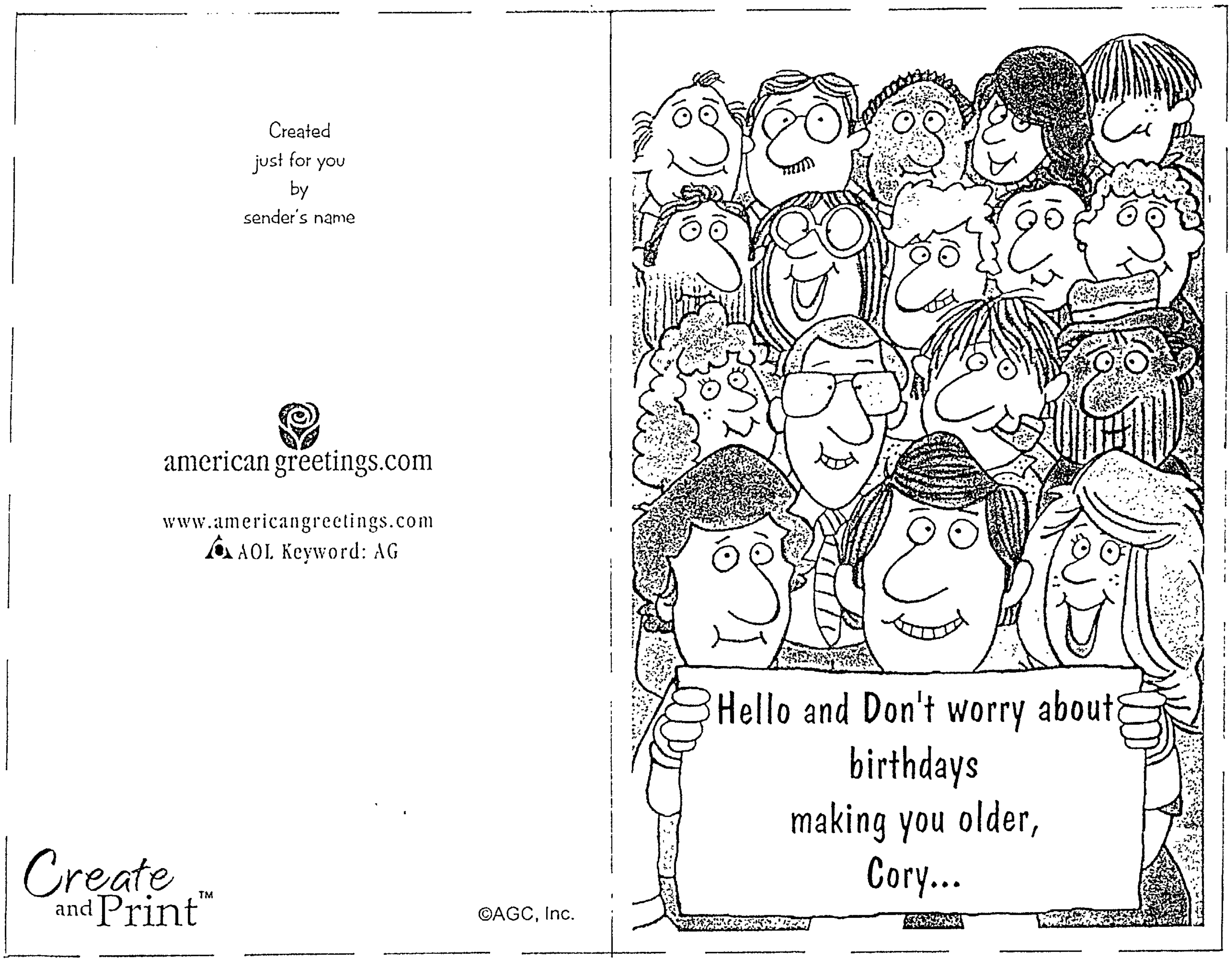
Point Size
20

Text Color
■

Text Alignment
Left
[✓ Center]
Right

Print

Fig. 5D


Created
just for you
by
sender's name

americangreetings.com

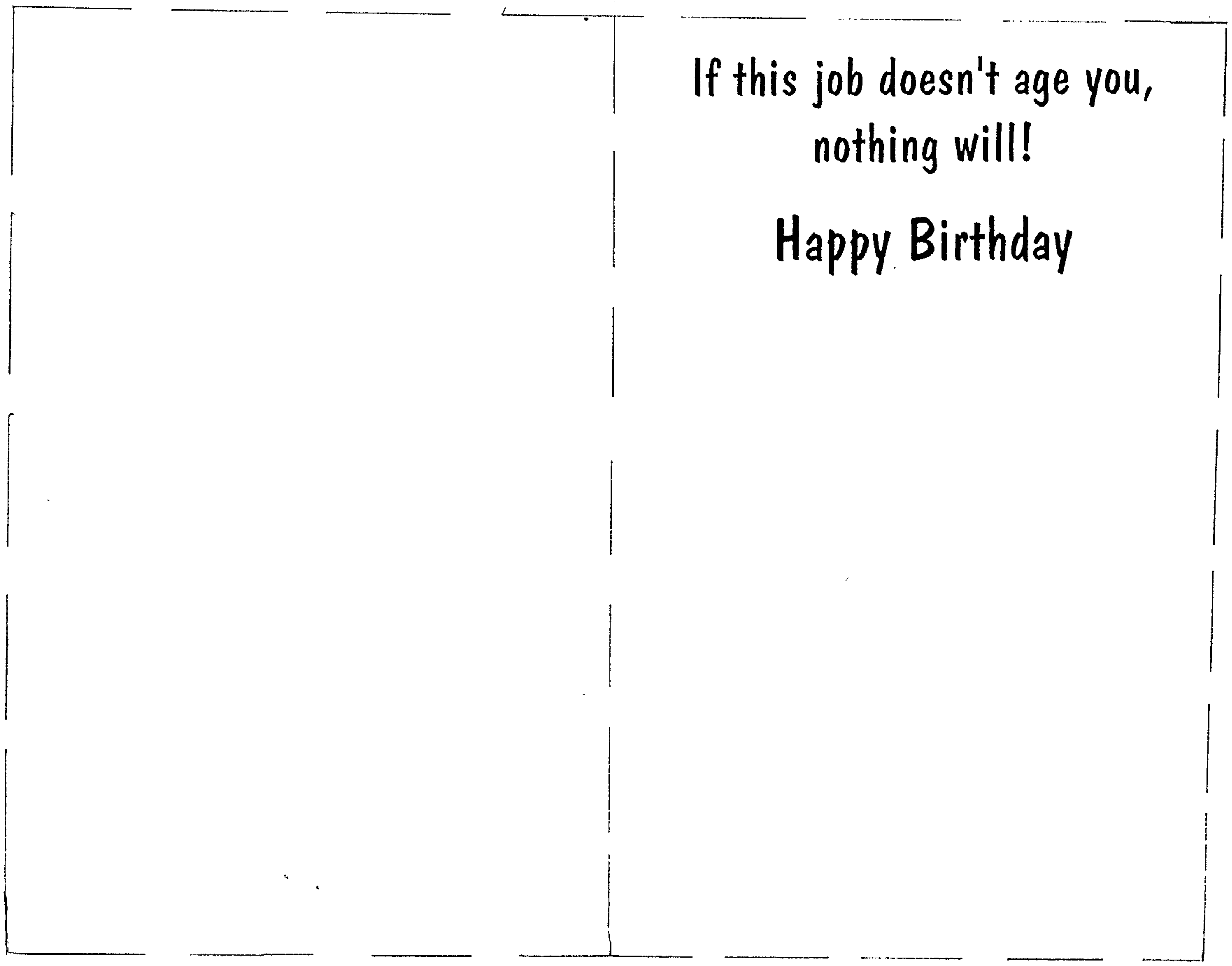
www.americangreetings.com
AOL Keyword: AG

Create and Print™

©AGC, Inc.

Hello and Don't worry about
birthdays
making you older,
Cory...

Fig. 6A

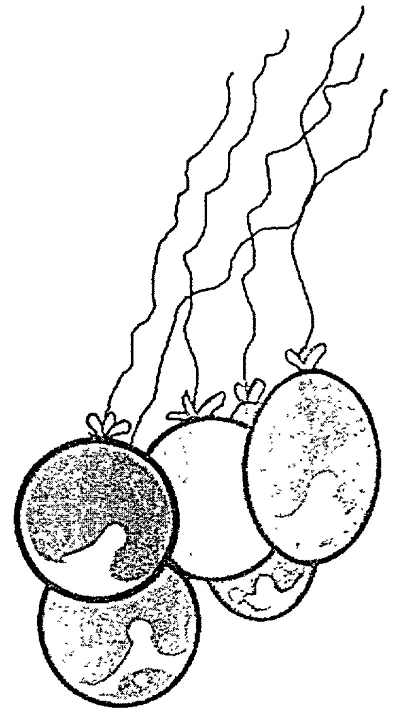

If this job doesn't age you,
nothing will!

Happy Birthday

Fig. 6B

Terry Smith is now
a published author!
Watch for Terry's book,
City Life,
to be in the stores
this Fall!

We're proud of you,
Terry!



Created
just for you
by
sender's name

americangreetings.com

www.americangreetings.com
AOL Keyword: AG

Create and Print™

©AGC, Inc

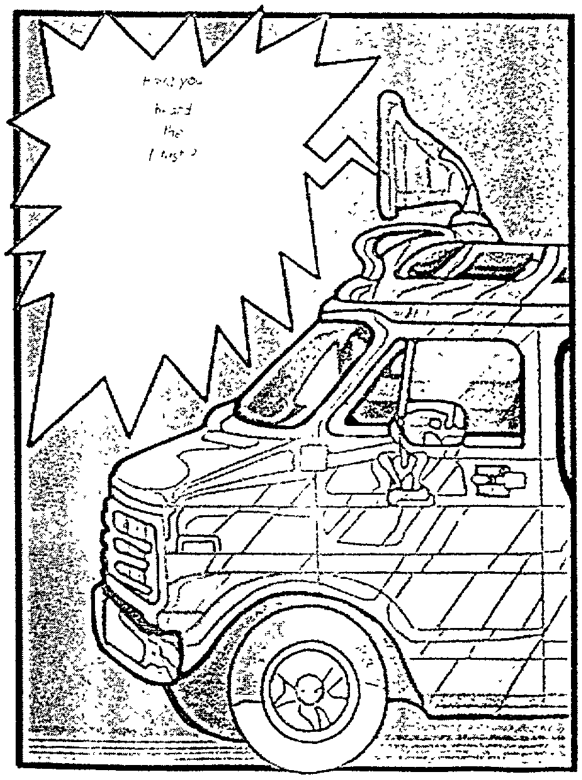


Fig. 7

Docket No. 74218/05085

DECLARATION FOR PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address, and citizenship are as stated below next to my name.

I believe that I am the original, first and sole inventor (if only one name is listed below), or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed, and for which a patent is sought on the invention entitled:

ON-LINE SYSTEM FOR CREATING A PRINTABLE PRODUCT

the specification of which is attached hereto, unless the following box is checked:

___ was filed on ______________________, 20___, as United States Application Number or PCT International Application Number ______________________.

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, §1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, §§119(a) - (d) or §365(b) of any foreign application(s) for patent or inventor's certificate, or §365(a) of any PCT International application which designated at least one country other than the United States, listed below and have also identified below any foreign application for patent or inventor's certificate, or PCT International application having a filing date before that of the application on which priority is claimed:

NONE

I hereby claim the benefit under Title 35, United States Code, §119(e) of any United States provisional application(s) listed below:

NONE

I hereby claim the benefit under Title 35, United States Code, §120 of any United States application(s), or §365(c) of any PCT International application designating the United States, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT International application in the manner provided by the first paragraph of Title 35, United States Code, §112, I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, §1.56 which became available between the filing date of the prior application and the national or PCT International filing date of this application:

NONE

I hereby appoint the following registered attorney(s) and/or agent(s) to prosecute this application and to transact all business in the Patent and Trademark Office connected therewith:

Alan J. Ross, Reg. No. 33,767
John X. Garred, Reg. No. 31,830
Michael A. Jaffe, Reg. No. 36,326
Susan L. Mizer, Reg. No. 38,245
Todd R. Tucker, Reg. No. 40,850
James C. Scott, Reg. No. 35,351
Jay P. Ryan, Reg. No. 37,064

Direct all telephone calls to	:	Michael A. Jaffe
at telephone number	:	(216) 696-3394
Direct all correspondence to	:	Michael A. Jaffe

ARTER & HADDEN LLP
1100 Huntington Building
925 Euclid Avenue
Cleveland, Ohio 44115-1475

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under 18 U.S.C. 1001 and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of sole or first inventor: Bruce T. Petro

Inventor's signature	: ______________________
Date	: ______________________
Residence	: Avon Lake, Ohio
Citizenship	: United States of America
Post Office Address	: 519 Marbrook Lane, Avon Lake, Ohio 44012

Full name of second inventor: Andrew Cohen

Inventor's signature	: ______________________
Date	: ______________________
Residence	: Colleyville, Texas
Citizenship	: United States of America
Post Office Address	: 3503 Pembrook Parkway South, Colleyville, Texas 76034

Full name of third inventor: Jason Sulak

Inventor's signature	: ______________________
Date	: ______________________
Residence	: Colleyville, Texas
Citizenship	: United States of America
Post Office Address	: 3301 Middleton Way, Colleyville, Texas 76034

APPENDIX

Inventors: Bruce T. Petro, Andrew Cohen and Jason Sulak

Title: ON-LINE SYSTEM FOR CREATING A PRINTABLE PRODUCT

```
                              long,
                              scKeyRecord&,
                              Bool );

    void        DoForwardDelete( long&,
                                 long&,
                                 scSpecRun&,
                                 long,
                                 scKeyRecord&,
                                 Bool );

    void        DoDiscHyphen( long&,
                              long&,
                              scSpecRun&,
                              long,
                              scKeyRecord&,
                              Bool );

    void        DoFixSpace( long&,
                            long&,
                            scSpecRun&,
                            long,
                            scKeyRecord&,
                            Bool );

    void        DoCharacter( long&,
                             long&,
                             scSpecRun&,
                             long,
                             scKeyRecord&,
                             Bool );

#endif

};

/* ================================================================== */

long        TXTStartWord( CharRecordP, long, int eleminateLeadingSpaces );
long        TXTEndWord( CharRecordP, long );
long        TXTStartSelectableWord( CharRecordP, long );
long        TXTEndSelectableWord( CharRecordP, long );

MicroPoint  UnivStringWidth( stUnivString&, MicroPoint[], TypeSpec& );

#ifdef jis4051
Bool        TXTSameRenMoji( CharRecordP start, CharRecordP ch1, CharRecordP ch2 );
#else
inline Bool TXTSameRenMoji( CharRecordP, CharRecordP, CharRecordP )
{
    return false;
}
#endif

/* ================================================================== */
/* ================================================================== */
/* ================================================================== */

class scContUnit : public scTBObj {
    scDECLARE_RTTI;
public:

                        // use this to allocate new content units where the content unit
                        // has been overridden on the outside.
    static scContUnit* Allocate( TypeSpec&      spec,
                                 scContUnit*    cu = 0,
                                 long           ct = 0 );

                    scContUnit();
                    scContUnit( TypeSpec& spec,
                                scContUnit* cu = 0,
                                long ct = 0 );
    void        WriteText( scSpecRun&,
                           Bool,
                           APPCtxPtr    ctxPtr,
                           IOFuncPtr    writeFunc,
                           int          charset = 0 );

    long        ReadAPPText( scSpecRun&, stTextImportExport& );
    void        WriteAPPText( scSpecRun&, stTextImportExport& );

    long        GetContentSize( void ) const
                    {
                        return fNumItems - 1;
                    }
    void        SetContentSize( long );
    long        ExternalSize( void ) const;

    void        Read( APPCtxPtr, IOFuncPtr );
    void        Write( APPCtxPtr, IOFuncPtr );

    virtual ElementPtr  Lock( void );
    virtual void        Unlock( void );
    void                Validate( void ) const;


private:
    void        CopyChars( CharRecordP, long, long );

#ifdef _RUBI_SUPPORT
    void        DoBackSpace( long&,
                             long&,
                             scSpecRun&,
                             scRubiArray*,
                             long,
                             scKeyRecord&,
                             Bool );

    void        DoForwardDelete( long&,
                                 long&,
                                 scSpecRun&,
                                 scRubiArray*,
                                 long,
                                 scKeyRecord&,
                                 Bool );

    void        DoDiscHyphen( long&,
                              long&,
                              scSpecRun&,
                              scRubiArray*,
                              long,
                              scKeyRecord&,
                              Bool );

    void        DoFixSpace( long&,
                            long&,
                            scSpecRun&,
                            scRubiArray*,
                            long,
                            scKeyRecord&,
                            Bool );

    void        DoCharacter( long&,
                             long&,
                             scSpecRun&,
                             scRubiArray*,
                             long,
                             scKeyRecord&,
                             Bool );


#else
    void        DoBackSpace( long&,
                             long&,
                             scSpecRun&,
                // copy the contents from startOffset to endOffset into the
                // arg scCharArray and then remove them
    void        Cut( scCharArray&, long, long );

                // paste the contents of the arg scCharArray into the character array
                // at the indicated array
    void        Paste( scCharArray&, long startOffset );

    int         FindString( const stUnivString&, const SearchState&, int32, int32, int32& );
    int         ReplaceToken( const stUnivString&, int32, int32& );
    int         GetToken( stUnivString&, int32, int32 ) const;

    void        Insert( const CharRecordP, long, long );
    void        Insert( const UCS2*, long, long );
    int         Insert( const stUnivString&, int32, int32 );

    void        CopyChars( UCS2*, long, long );


                // transform the indicated characters using the type of
                // transformation passed in, ususally for making
                // alternate characters
    void        Transform( long         startOffset,
                           long         endOffset,
                           eChTranType  trans,
                           int          numChars );

    void        Retabulate( scSpecRun&  specRun,
                            long        start,
                            long        end,
                            TypeSpec    changedSpec,
                            long        charSize );

    void        RepairText( scSpecRun&,
                            long        offset1,
                            long        offset2 );

    void        SelectWord( long    offset,
                            long&   startWord,
                            long&   endWord );

#ifdef _RUBI_SUPPORT
    void        CharInsert( long&,
                            scSpecRun&,
                            scRubiArray*,
                            long,
                            scKeyRecord&,
                            Bool,
                            TypeSpec );
#else
    void        CharInsert( long&,
                            scSpecRun&,
                            long,
                            scKeyRecord&,
                            Bool,
                            TypeSpec );
#endif

    void        WordSpaceInfo( long, MicroPoint& );

    void        CharInfo( scSpecRun&,
                          long,
                          UCS2&,
                          ulong&,
                          MicroPoint&,
                          TypeSpec&,
                          eUnitType& );

    long        ReadText( scSpecRun&,
                          APPCtxPtr     ctxPtr,
                          IOFuncPtr     readFunc,
                          int           charset = 0 );
```

```
        // that is used to insure correct update with mono-spaced
        // characters

                    scStreamChangeInfo( ) :
                        fColumn( 0 ),
                        fPara( 0 ),
                        fOffset( 0 ),
                        fLength( 0 ){}

    void            Set( scColumn* col, scContUnit* para, long offset, long len ) {
                        fColumn     = col,
                        fPara       = para,
                        fOffset     = offset,
                        fLength     = len;
                    }

    scColumn*       GetColumn( void ) const     { return fColumn; }
    scContUnit*     GetPara( void ) const       { return fPara; }
    long            GetOffset( void ) const     { return fOffset; }

    long            GetLength( void ) const     { return fLength; }
    void            SetLength( long len )       { fLength = len; }

private:
    scColumn*       fColumn;
    scContUnit*     fPara;
    long            fOffset;
    long            fLength;

};

/* ================================================================== */

class PrevParaData {
public:
                PrevParaData()
                    {
                        Init();
                    }
    void        Init( void )
                    {
                        lastLineH = 0;
                        lastSpec.clear();
                     }
    scTextline* lastLineH;
    TypeSpec    lastSpec;

};

/* ================================================================== */

class scCharArray : public scHandleArray {
    scDECLARE_RTTI;
public:
                scCharArray() :
                        scHandleArray( sizeof( CharRecord ) )
                        {
                            CharRecord ch( 0, 0 );
                            AppendData( (ElementPtr)&ch );      // add null terminator
                        }


    virtual int     IsEqual( const scObject& ) const;

    UCS2        GetCharAtOffset( long offset ) const
                    { return (((CharRecordP)GetMem()) + offset )->character; }

    void        RemoveBetweenOffsets( long startOffset, long endOffset );

                // copy the contents from startOffset to endOffset into the
                // arg scCharArray
    void        Copy( scCharArray&, long startOffset, long endOffset ) const;
```

```
/*****************************************************************************

    File:       SCPARAGR.H


    $Header: /Projects/Toolbox/ct/SCPARAGR.H 3     5/30/97 8:45a Wmanis $

    Contains:   Method/Function interface to class of paragraph

    Written by: Manis


    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

*****************************************************************************/

#ifndef _H_SCPARAGR
#define _H_SCPARAGR

#ifdef SCMACINTOSH
#pragma once
#endif

#include "sctbobj.h"
#include "sccharex.h"
#include "scspcrec.h"
#include "scmemarr.h"

///////////////////////////////////////////////////////////////////////////////
/////////////////////////////// FORWARD REFERENCES ///////////////////////////
///////////////////////////////////////////////////////////////////////////////

#ifdef _RUBI_SUPPORT
class scRubiArray;
#endif

class scColumn;
class scCOLRefData;
class scSpecRecord;
class scMuPoint;
class scAnnotation;
class scLEADRefData;
class stTextImportExport;
class scTypeSpecList;
class scSpecLocList;
class scTextline;

/* ================================================================== */
// events that the reformatter returns

typedef enum eReformatEvents {
    eNoReformat,                // no reformatting was performed
    eNormalReformat,            // normal reformatting event
    eRebreak,                   // rebreak the paragraph, probably for widow/orphan control
    eOverflowGeometry,          // more text than columns
    eOverflowContent            // more columns than text
} eRefEvent;

/* ================================================================== */


class scStreamChangeInfo {
public:
        // these are the paragraph and offset of character insertion
```

```
/* ================================================================== */

#ifndef SCmemset        // we are in a 16 bit world

void scFar*  scFar scCDecl SCmemset( void scFar*    ptr,
                                     int            val,
                                     long           len )
{
    return _fmemset( ptr, val, (size_t)len );
}

/* ================================================================== */

void scFar*  scFar scCDecl SCmemmove( void scFar*       dst,
                                      const void scFar* src,
                                      long              len )
{
    return _fmemmove( dst, src, (size_t)len );
}

/* ================================================================== */

void scFar*  scFar scCDecl SCmemcpy( void scFar*       dst,
                                     const void scFar* src,
                                     long              len )
{
    return _fmemcpy( dst, src, (size_t)len );
}

/* ================================================================== */

int scFar scCDecl SCmemcmp( const void scFar*   p1,
                            const void scFar*   p2,
                            long                len )
{
    return _fmemcmp( p1, p2, (size_t)len );
}

#endif
/* ================================================================== */
```

```
        ulong   sz = MemSize( obj );

        hnd = MEMAllocHndDebug( sz, filename, line );

        try {
            void*   srcP = MemLock( obj );
            void*   dstP = MemLock( hnd );
            SCmemcpy( dstP, srcP, sz );
        }

        catch ( ... ) {
            MemUnlock( hnd );
            MemUnlock( obj );
            throw;
        }

        MemUnlock( hnd );
        MemUnlock( obj );

#else

        ulong   sz = _msize( obj ) - sizeof( MacHandle );

        hnd = MEMAllocHndDebug( sz, filename, line );

        try {
            void*   srcP = MEMLockHnd( obj );
            void*   dstP = MEMLockHnd( hnd );
            SCmemcpy( dstP, srcP, sz );
        }

        catch ( ... ) {
            MEMUnlockHnd( hnd );
            MEMUnlockHnd( obj );
        }

        MEMUnlockHnd( hnd );
        MEMUnlockHnd( obj );

#endif
    }
    else
        hnd = NULL;
    raise_if( !hnd, scERRmem );
    memRecordTrackInfo( hnd, filename, line );
    return hnd;
}

/* ========================================================================== */

#endif /* SCDEBUG */

/* ========================================================================== */

scAutoUnlock::scAutoUnlock( scMemHandle hnd )
    : fHandle(hnd)
{
#if useSMARTHEAP
    MemLock( fHandle );
#else
    MEMLockHnd( fHandle );
#endif
}

scAutoUnlock::~scAutoUnlock()
{
#if useSMARTHEAP
    MemUnlock( fHandle );
#else
    MEMUnlockHnd( fHandle );
#endif
}
```

```
#if useSMARTHEAP
    if ( !*obj )
        ptr = MEMAllocPtrDebug( reqSize, file, line );
    else
        ptr = _dbgMemReAllocPtr( *obj, reqSize, MEM_RESIZEABLE, file, line );
#else
    ptr = realloc( *obj, reqSize );
#endif

    raise_if( !ptr, scERRmem );
    return *obj = ptr;
}

/* ======================================================================== */

scMemHandle MEMResizeHndDebug( scMemHandle  obj,
                               ulong        reqSize,
                               const char*  file,
                               int          line )
{
    CLoanApp    loanApp;

#if useSMARTHEAP
    if ( !obj )
        obj = MEMAllocHndDebug( reqSize, file, line );
    else
        obj = _dbgMemReAlloc( obj, reqSize, MEM_RESIZEABLE, file, line );
#else
    obj = (scMemHandle)realloc( obj, reqSize + sizeof( MacHandle ) );
#endif

    return obj;
}

/* ======================================================================== */

void *MEMDupPtrDebug( void *obj, const char *filename, int line )
{
    CLoanApp    loanApp;
    void        *ptr;

    if ( !RandomFailure() ) {
#if useSMARTHEAP
        ulong       sz = MemSizePtr( obj );

        ptr = MEMAllocPtrDebug( sz, filename, line );
        raise_if( !ptr, scERRmem );
        SCmemcpy( ptr, obj, sz );
#else
        ulong       sz = _msize( obj );

        ptr = MEMAllocPtrDebug( sz, filename, line );
        raise_if( !ptr, scERRmem );
        SCmemcpy( ptr, obj, sz );
#endif
    }
    else
        ptr = NULL;
    raise_if( !ptr, scERRmem );
    memRecordTrackInfo(ptr, filename, line);
    return ptr;
}

/* ======================================================================== */

scMemHandle MEMDupHndDebug( scMemHandle obj, const char *filename, int line )
{
    CLoanApp    loanApp;
    scMemHandle hnd;

    if ( !RandomFailure() ) {

#if useSMARTHEAP
```

```
int gRandomFailure;                 // randomly fail memory allocations

static Boolean RandomFailure()
{
    if ( !gRandomFailure || !gStartUpCompleted )
        return false;

    if ( ( rand() % gRandomFailure ) )
        return false;
    else {
        SCDebugTrace( 0, scString( "RANDOM FAILURE %d\n" ), gRandomFailure );
        return true;
    }
}

/* ======================================================================== */

void* MEMAllocPtrDebug( ulong sz, const char *filename, int line )
{
    CLoanApp    loanApp;
    void        *ptr;

    raise_if( RandomFailure(), scERRmem );

#if useSMARTHEAP
    ptr = _dbgMemAllocPtr( GetPool( sz ), sz, 0, filename, line );
#else
    ptr = malloc( sz );
#endif

    raise_if( !ptr, scERRmem );
    memRecordTrackInfo(ptr, filename, line);

    return ptr;
}

/* ======================================================================== */

scMemHandle MEMAllocHndDebug( ulong sz, const char *filename, int line )
{
    CLoanApp    loanApp;
    scMemHandle hnd;

    raise_if( RandomFailure(), scERRmem );

#if useSMARTHEAP
    hnd = _dbgMemAlloc( GetHandlePool(), MEM_MOVEABLE | MEM_RESIZEABLE, sz, filename, line );
#else

    hnd = (scMemHandle)malloc( sizeof( MacHandle) + sz );

    MacHandle macHandle( hnd );

    *(MacHandle*)hnd = macHandle;

#endif

    raise_if( !hnd, scERRmem );
    memRecordTrackInfo( hnd, filename, line);

    return hnd;
}

/* ======================================================================== */

void* MEMResizePtrDebug( void**         obj,
                         ulong          reqSize,
                         const char*    file,
                         int            line )
{
    CLoanApp    loanApp;
    void        *ptr;
```

```
{
#if useSMARTHEAP
    return MemLock( hnd );
#else
    MacHandle* mh = (MacHandle*)hnd;
    return mh->Lock();
#endif

}

/* ========================================================================= */

void MEMUnlockHnd( scMemHandle hnd, int counted )
{
#if useSMARTHEAP
    MemUnlock( hnd );
#else
    MacHandle* mh = (MacHandle*)hnd;
    mh->Unlock();
#endif
}

/* ========================================================================= */

#if SCDEBUG > 1

/* ========================================================================= */

void    MEMValidate( void *ptr )
{
#if useSMARTHEAP
    MEM_POOL pool = PoolOfPtr( ptr );

    if ( pool ) {
        scAssert( MemPoolCheck( pool ) );
    }
#else
#endif
}

/* ========================================================================= */

void memDumpMetrics()
{
#if useSMARTHEAP
#elif useMACHACK
#endif
}

/* ========================================================================= */

inline void memRecordTrackInfo( void *ptr, const char *filename, int line )
{
#ifdef MEM_TRACK_ALLOC
    #if useSMARTHEAP
    #else
    #endif
#endif
}

/* ========================================================================= */

inline void memRecordTrackInfo( scMemHandle ptr, const char *filename, int line )
{
#ifdef MEM_TRACK_ALLOC
    #if useSMARTHEAP
    #else
    #endif
#endif
}

/* ========================================================================= */
```

```
    MacHandle macHandle( obj );

    *(MacHandle*)obj = macHandle;
#endif

    return obj;
}

/* ========================================================================== */

#endif /* !SCDEBUG */

/* ========================================================================== */

void MEMFreePtr( void *obj )
{
    if ( obj == 0 )
        return;

#if useSMARTHEAP
    MemFreePtr( obj );
#else
    free( obj );
#endif
}

/* ========================================================================== */

void MEMFreeHnd( scMemHandle obj )
{
    if ( obj == 0 )
        return;

#if useSMARTHEAP
    raise_if( MemLockCount( obj ), scERRmem );
    MemFree( obj );
#else
    free( obj );
#endif
}

/* ========================================================================== */

ulong MEMGetSizePtr( const void *obj )
{
    if ( obj == 0 )
        return 0;

#if useSMARTHEAP
    return MemSizePtr( (void*)obj );
#else
    return _msize( (void*)obj );
#endif
}

/* ========================================================================== */

ulong MEMGetSizeHnd( scMemHandle obj )
{
    if ( obj == 0 )
        return 0;

#if useSMARTHEAP
    return MemSize( obj );
#else
    return _msize( (void*)obj ) - sizeof( MacHandle );
#endif
}

/* ========================================================================== */

void *MEMLockHnd( scMemHandle hnd, int counted )
```

```
    hnd = MEMAllocHnd( sz );

    try {
        void*   srcP = MEMLockHnd( obj );
        void*   dstP = MEMLockHnd( hnd );
        SCmemcpy( dstP, srcP, sz );
    }

    catch( ... ) {
        MEMUnlockHnd( hnd );
        MEMUnlockHnd( obj );
    }

    MEMUnlockHnd( hnd );
    MEMUnlockHnd( obj );

#endif

    return hnd;
}

/* ========================================================================= */

void *MEMDupObj( void *obj )
{
    CLoanApp    loanApp;
    void        *ptr;
    ulong       sz = MEMGetSizePtr( obj );

    ptr = MEMAllocPtr( sz );
    raise_if( !ptr, scERRmem );
    SCmemcpy( ptr, obj, sz );
    return ptr;
}

/* ========================================================================= */

void* MEMResizePtr( void**  obj, ulong reqSize )
{
    CLoanApp    loanApp;
    void        *ptr;

#if useSMARTHEAP
    if ( !*obj )
        ptr = MEMAllocPtr( reqSize );
    else
        ptr = MemReAllocPtr( *obj, reqSize, MEM_RESIZEABLE );
#else
    if ( !*obj )
        ptr = malloc( reqSize );
    else
        ptr = realloc( *obj, reqSize );
#endif
    raise_if( !ptr, scERRmem );
    return *obj = ptr;
}

/* ========================================================================= */

scMemHandle MEMResizeHnd( scMemHandle obj, ulong reqSize )
{
    CLoanApp    loanApp;

#if useSMARTHEAP
    if ( !obj )
        obj = MEMAllocHnd( reqSize );
    else
        obj = MemReAlloc( obj, reqSize, MEM_RESIZEABLE );
#else
    if ( !obj )
        obj = MEMAllocHnd( reqSize );
    else
        obj = (scMemHandle)realloc( obj, reqSize + sizeof( MacHandle ) );
    CLoanApp    loanApp;
    scMemHandle hnd = 0;

#if useSMARTHEAP
    hnd = MemAlloc( GetHandlePool(), MEM_MOVEABLE | MEM_RESIZEABLE, sz );
#else
    hnd = (scMemHandle)malloc( sizeof( MacHandle) + sz );

    MacHandle macHandle( hnd );

    *(MacHandle*)hnd = macHandle;
#endif

    raise_if( !hnd, scERRmem );
    return hnd;
}

/* ======================================================================== */

//void *MEMAllocObj( ulong size )
//{
//  CLoanApp    loanApp;
//  void        *ptr;
//
//  ptr = GetMemManager().AllocObj( (size_t)size );
//  raise_if( !ptr, scERRmem );
//  return ptr;
//}

/* ======================================================================== */

void *MEMDupPtr( void *obj )
{
    CLoanApp    loanApp;
    void        *ptr;
    ulong       sz = MEMGetSizePtr( obj );

    ptr = MEMAllocPtr( sz );
    raise_if( !ptr, scERRmem );
    SCmemcpy( ptr, obj, sz );
    return ptr;
}

/* ======================================================================== */

scMemHandle MEMDupHnd( scMemHandle obj )
{
    CLoanApp    loanApp;
    scMemHandle hnd;

#if useSMARTHEAP
    ulong   sz = MemSize( obj );

    hnd = MEMAllocHnd( sz );

    try {
        void*   srcP = MemLock( obj );
        void*   dstP = MemLock( hnd );
        SCmemcpy( dstP, srcP, sz );
    }

    catch( ... ) {
        MemUnlock( hnd );
        MemUnlock( obj );
    }

    MemUnlock( hnd );
    MemUnlock( obj );

#else

    ulong   sz = MEMGetSizePtr( obj );
```

```
int                 gStartUpCompleted;

// NOTE: To understand this you should be aware of the Macintosh memory
// management as well as the handling of memory in the CApplication class.
// Read the TCL description of the CApplication class and how it handles
// the rainy day fund.
// The stack object CLoanApp tells the application that we can fail this
// memory request. We will assume that all other requests cannot fail.
// That means we must have sufficient memory to service the request.


    // this should really be a CStackObject - unfortunatley the chicken/egg
    // problem arises because the init of tcExceptContext calls these routines
    // and CStackObject relies upon tcExceptContext already existing.
    // The reason we would like it to be a stack object is that if we
    // throw and exception this would reset the memory requests properly.
    // To reset the the memory request flags in the application I will
    // set them when we ignore the exception at the top of the event loop.

class CLoanApp {
public:
    CLoanApp();
    ~CLoanApp();
private:
};


CLoanApp::CLoanApp()
{
}

CLoanApp::~CLoanApp()
{
}

#else

int                 gStartUpCompleted = true;

#define CLoanApp
#define loanApp

#endif

/* ======================================================================== */
/* ======================================================================== */
/* ======================================================================== */

#if SCDEBUG < 2

/* ======================================================================== */

void *MEMAllocPtr( ulong sz )
{
    CLoanApp    loanApp;
    void        *ptr;

#if useSMARTHEAP
    ptr = MemAllocPtr( GetPool( sz ), sz, 0 );
#else
    ptr = malloc( sz );
#endif

    raise_if( !ptr, scERRmem );
    return ptr;
}

/* ======================================================================== */

scMemHandle MEMAllocHnd( ulong sz )
{
        scStrcat( buf, scString( "MEM_VAR_MOVEABLE_BLOCK " ) );
    if ( info->type & MEM_VAR_FIXED_BLOCK )
        scStrcat( buf, scString( "MEM_VAR_FIXED_BLOCK " ) );
    SCDebugTrace( 0, scString( "MEM_BLOCK_TYPE %s\n" ), buf );

    SCDebugTrace( 0, scString( "pagesize %d\n" ), info->pageSize );
    SCDebugTrace( 0, scString( "floor %lu\n" ), info->floor );
    SCDebugTrace( 0, scString( "ceiling %lu\n" ), info->ceiling );
    SCDebugTrace( 0, scString( "flags 0x%08x\n" ), info->flags );


}
#endif

/* ====================================================================== */

void MEMFini()
{
    register        i;

#if SCDEBUG > 1
    MEM_POOL_INFO   info;
    SCDebugTrace( 0, scString( "\n\nMemFini: BEGIN\n" ) );
#endif

#if MEM_DEBUG
    dbgMemSetDefaultErrorOutput( DBGMEM_OUTPUT_CONSOLE, "leakage.out" );
#endif

    for ( i = 0; i < numPools; i++ ) {

        SCDebugTrace( 0, scString( "Free MemPool - start %d\n" ), i );
#if SCDEBUG > 1
        MemPoolCheck( pools[i].fPool );
        MemPoolInfo( pools[i].fPool, 0, &info );
        dbgMemFormatPoolInfo( &info );
#endif

#if MEM_DEBUG
        scAssert( dbgMemReportLeakage( pools[i].fPool, 1, UINT_MAX  ) );
#endif

        scAssert( MemPoolFree( pools[i].fPool ) ), pools[i].fPool = 0;

        SCDebugTrace( 0, scString( "Free MemPool - end %d\n\n\n" ), i );
    }

#if SCDEBUG > 1
    SCDebugTrace( 0, scString( "MemFini: DONE\n" ) );
#endif
}

/* ====================================================================== */
#else
/* ====================================================================== */

void MEMInit( scPoolInfo [] )
{
}

/* ====================================================================== */

void MEMFini()
{
}

#endif

/* ====================================================================== */
/* ====================================================================== */
/* ====================================================================== */
#ifdef SCMACINTOSH
```

```
    register i;

    for ( i = 0; i < numPools; i++ ) {
        if ( pools[i].fBlockSize && pools[i].fBlockSize == size )
            return pools[i].fPool;
        else
            return pools[i].fPool;
    }
    return 0;
}


/* ======================================================================= */

inline MEM_POOL PoolOfPtr( void* ptr )
{
    MEM_POOL_INFO   info;

    if ( MemPoolInfo( 0, ptr, &info ) )
        return info.pool;
    else
        return 0;
}

/* ======================================================================= */

inline int CountPools( scPoolInfo infoPools[] )
{
    register i;

    for ( i = 0; infoPools[i++].fBlockSize; )
        ;

    return i;
}

/* ======================================================================= */

void MEMInit( scPoolInfo infoPools[] )
{
    register i;

    pools = infoPools;
    numPools = CountPools( pools );

    for ( i = 0;  i < numPools; i++ ) {
        if ( pools[i].fBlockSize ) {
            pools[i].fPool = MemPoolInitFS( pools[i].fBlockSize,
                                            1024,
                                            MEM_POOL_DEFAULT );
            raise_if( pools[i].fPool == 0, scERRmem );
        }
        else {
            pools[i].fPool = MemPoolInit( MEM_POOL_DEFAULT );
            raise_if( pools[i].fPool == 0, scERRmem );
        }
    }
}

/* ======================================================================= */

#if SCDEBUG > 1

void dbgMemFormatPoolInfo( MEM_POOL_INFO* info )
{
    scChar buf[256];

    SCDebugTrace( 0, scString( "MEM_POOL_INFO\n" ) );

    scStrcpy( buf, scString( "" ) );
    if ( info->type & MEM_FS_BLOCK )
        scStrcat( buf, scString( "MEM_FS_BLOCK " ) );
    if ( info->type & MEM_VAR_MOVEABLE_BLOCK )
```

```
/*************************************************************************

	File:		MEM.C

	$Header: /Projects/Toolbox/ct/SCMEM.CPP 2     5/30/97 8:45a Wmanis $

	Contains:	Memory management routines based on our own heap managers

	Written by: Sealy

	Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
	All rights reserved.

	This notice is intended as a precaution against inadvertent publication
	and does not constitute an admission or acknowledgment that publication
	has occurred or constitute a waiver of confidentiality.

	Composition Toolbox software is the proprietary
	and confidential property of Stonehand Inc.

**************************************************************************/

#include "scmem.h"

#if !useSMARTHEAP

#include <malloc.h>

struct MacHandle {
	const void*	fBlock;
	int			fCount;

			MacHandle( scMemHandle  ptr ) :
						fBlock( (char*)ptr + sizeof( MacHandle ) ),
						fCount( 0 ){}

	void*	Lock( void ) { scAssert( fCount >= 0 ); fCount++; return (void*)fBlock; }
	void	Unlock( void ) { scAssert( fCount > 0 ); --fCount; }
};

#endif

#include "scexcept.h"
#include <string.h>

#if SCDEBUG > 1
	#include <stdlib.h>	// for rand
#endif

/* ======================================================================== */
/* ======================================================================== */
/* ======================================================================== */

#if useSMARTHEAP > 0

static MEM_POOL		hndPool;

static int			numPools;
static scPoolInfo*	pools;

/* ======================================================================== */

inline MEM_POOL GetHandlePool( void )
{
	return pools[ numPools - 1 ].fPool;
}

/* ======================================================================== */

inline MEM_POOL GetPool( size_t size )
{
```

```
/*=================================================================

    File:       SCMACINT.H

    $Header: /Projects/Toolbox/ct/SCMACINT.H 2     5/30/97 8:45a Wmanis $

    Contains:   Defines for MacIntosh/MPW compile

    Written by:

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

=================================================================*/



#ifndef _H_SCMACINT
#define _H_SCMACINT

/*
 *  Header configuration.  To determine if we are using universal headers load types.h and then
 *  look to see if it has included the universal headers <ConditionalMacros.h> file.
 */
#include <Types.h>

#ifdef __CONDITIONALMACROS__
#define USEUNIVERSALHEADERS 1
#else
#define USEUNIVERSALHEADERS 0
#endif

#if USEUNIVERSALHEADERS && !defined(USESROUTINEDESCRIPTORS)
#define USESROUTINEDESCRIPTORS 1
#endif

/* SYSTEM INCLUDES */
#include "StdDef.h"


//#include <OSUtils.h>
//#include <Events.h>
//#include <limits.h>
#include <string.h>
//#include <math.h>




    // memory model stuff - for intel only
#define scNEAR
#define scFAR
#define SChuge

//volatile is not supported by MPW
#define volatile


#define SCTickCount()           (TickCount())
#define SCSysBeep(duration)     (SysBeep((int)(duration)))

#endif /* _H_SCMACINT.H */
```

```
uchar*          BufSet_REAL( uchar          rbuf[12],
                             REAL           r,
                             eByteOrder     desiredByteOrder  );

const uchar*    BufGet_REAL( const uchar    rbuf[12],
                             REAL&          pr,
                             eByteOrder     byteOrder );


                    // the follow are not good for
                    // writing out alot of data, but for a long
                    // here are there they are goo
void            ReadLong( long&,
                          APPCtxPtr,
                          IOFuncPtr,
                          eByteOrder );

                    // a quick way of writing out a long
void            WriteLong( long,
                           APPCtxPtr,
                           IOFuncPtr,
                           eByteOrder );

void            ReadBytes( uchar*,
                           APPCtxPtr,
                           IOFuncPtr,
                           long );

void            WriteBytes( const uchar*,
                            APPCtxPtr,
                            IOFuncPtr,
                            long );

#endif
```

```
/*****************************************************************

    File:       pfileio.h

    $Header: /Projects/Toolbox/ct/SCFILEIO.H 2     5/30/97 8:45a Wmanis $

    Contains:   Independent byte order calls.

    Written by: Coletti

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

*****************************************************************/

#ifndef _H_PFILEIO
#define _H_PFILEIO

#include "sctypes.h"


typedef enum eByteOrders {
    kNoOrder        = 0,
    kIntelOrder     = 1,
    kMotorolaOrder  = 2
} eByteOrder;


typedef uchar   REALBUF[12];
typedef uchar   ByteOrderStr[8];


#define kShortBufSize   2
#define kLongBufSize    4


uchar*          BufSet_byteorder( uchar[] );
const uchar*    BufGet_byteorder( const uchar[], short* );


uchar*          BufSet_char( uchar*         dstbuf,
                             const uchar*   srcbuf,
                             size_t         bytes,
                             eByteOrder     desiredByteOrder );

const uchar*    BufGet_char( const uchar*   srcbuf,
                             uchar*         dstbuf,
                             size_t         bytes,
                             eByteOrder     byteOrder );

uchar*          BufSet_short( uchar         sbuf[2],
                              ushort        s,
                              eByteOrder    desiredByteOrder  );

const uchar*    BufGet_short( const uchar   sbuf[2],
                              ushort&       ps,
                              eByteOrder    byteOrder );

uchar*          BufSet_long( uchar          pbuf[4],
                             ulong          l,
                             eByteOrder     desiredByteOrder  );

const uchar*    BufGet_long( const uchar    lbuf[4],
                             ulong&         pl,
                             eByteOrder     byteOrder );
```

```
/***************************************************************************

    File:       SCGLOBDA.C

    $Header: /Projects/Toolbox/ct/SCGLOBDA.CPP 2     5/30/97 8:45a Wmanis $

    Contains:   Global data, which should be gone soon!

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

***************************************************************************/

#include "scexcept.h"

#include <string.h>
#include "scmem.h"

#include "scglobda.h"
#include "scparagr.h"
#include "sccolumn.h"
#include "sctextli.h"

scDEFINE_RTTI( scTBObj, scObject );
scDEFINE_RTTI( scColumn, scTBObj );
scDEFINE_RTTI( scTextline, scTBObj );
scDEFINE_RTTI( scContUnit, scTBObj );

scDEFINE_ABSTRACT_RTTI( scAbstractArray, scObject );
scDEFINE_RTTI( scHandleArray, scAbstractArray );
scDEFINE_RTTI( scMemArray, scAbstractArray );
scDEFINE_RTTI( scCharArray, scHandleArray );


char *SCS_Copyright  = "Copyright (c) 1988-1994 Stonehand Inc. All rights reserved.";

BreakStruct          gbrS;
GlobalColumnStruct   ggcS;
scStreamChangeInfo   gStreamChangeInfo;

Bool                 gHiliteSpaces;  // hilite trailing spaces at the end of a line

long    scLogUnitsPerPixel = 20;
```

```
    MicroPoint          totalTrailingSpace;

    long                theLineCount;

    Bool                firstGlue;
    Bool                firstBox;
    Bool                allowHyphens;
    Bool                allowJustification;

    Bool                fNoStartline;         /* true if previous char was        */
                                              /* starting punctuation             */
    MicroPoint          fLastHangable;        /* width of last character that was hangable */
    short               numTargetChars;       /* num target chars rubi applied to */

    short               lineHyphenated;

        /* this the setting for the line based upon
         * the first spec found on the line or a quad
         * character
         */
    eTSJust             effectiveRag;

        /* if the column has horz flex we
         * fit all the line flush left and
         * then reposition all the lines
         */
    eTSJust             colShapeRag;

    scColumn                *theBreakColH;

    DropCapInfo         dcInfo;
    MicroPoint          dcLastBaseline;

        /* true if this line contains a drop cap */
    Bool                dcSet;

        /* we found a character indent char on this line */
    Bool                foundCharIndent;
};

/***********************************************************************/

class GlobalColumnStruct {
public:
        GlobalColumnStruct()
                {
                }
        ~GlobalColumnStruct()
                {
                }
    TypeSpec        defaultSpec;
        /* this is the current column we are breaking in */
    scColumn*       theActiveColH;
};

/***********************************************************************/

extern BreakStruct          gbrS;
extern GlobalColumnStruct   ggcS;
extern scStreamChangeInfo   gStreamChangeInfo;

extern Bool                 gHiliteSpaces;  // hilite trailing spaces at the end of a line

#endif /* _H_SCGLOBDA */
```

```
                        scMaxLineVals() :
                            fSpecRec( 0 ),
                            fOblique( 0 ) {}

    void                Init( void )
                        { fSpecRec = 0; fMaxLead.Init( scFlowDir( eRomanFlow ) );
                          fMaxInkExtents.Set( 0, 0, 0, 0 ); fOblique = 0; }
    scSpecRecord*       fSpecRec;
    scLEADRefData       fMaxLead;
    scXRect             fMaxInkExtents;
    scAngle             fOblique;
};


/* ========================================================== */

enum eBreakEvent {
    start_of_line,
    in_line,
    measure_exceeded,
    end_of_stream_reached
};

typedef eBreakEvent (*BrFunc)( void );

class BreakStruct {
public:
                        BreakStruct();
                        ~BreakStruct();

    void                Init();

    BrFunc*             breakMach;
    CandBreak*          candBreak;

        // CURRENT BREAK POINT STATE
    CandBreak           cB;

    scMemHandle         brkLineValsH;
            // a list of max line vals for each spec on the line */
    scMaxLineVals*      fMaxLineVals;
            // zero this and make sure it stays that way */
    scMaxLineVals       fZeroMaxLineVals;

    CharRecordP         gStartRec;

    TypeSpec            pspec_;

    scSpecRecord*       theSpecRec;

    MicroPoint          tmpMinGlue;
    MicroPoint          tmpOptGlue;
    MicroPoint          tmpMaxGlue;
    GlyphSize           letterSpaceAdj;

    MicroPoint          originalMeasure;
    MicroPoint          desiredMeasure;
    MicroPoint          hyphenationZone;

                        /* length of last line set, for ragged setting */
    MicroPoint          lastLineLen;
    GlyphSize           justSpace;
    MicroPoint          theLineOrg;

        /* space set by character indent */
    MicroPoint          charIndent;
    MicroPoint          minRelPosition;

        /* we need local values of this in case
         * the spec changes on the line
         */
    MicroPoint          brkLeftMargin;
    MicroPoint          brkRightMargin;
```

```
/*****************************************************************

    File:       SCGLOBDA.H

    $Header: /Projects/Toolbox/ct/Scglobda.h 2     5/30/97 8:45a Wmanis $

    Contains:   Global data.

    Written by: Lucas

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

*****************************************************************/

#ifndef _H_SCGLOBDA
#define _H_SCGLOBDA

#ifdef SCMACINTOSH
    #pragma once
#endif

#include "sctypes.h"
#include "scselect.h"
#include "scsetjmp.h"
//#include "scvalue.h"
#include "scspcrec.h"
#include "screfdat.h"
#include "scparagr.h"


/*********************************************************************/
/* FOR USE IN THE LINE BREAKER */

class CandBreak {
public:
    long    breakCount;     /* the count */
    long    startCount;     /* stream count at start of line */
    long    streamCount;    /* stream count from start of this line */
    ushort      wsSpaceCount;   /* # of inter-word spaces at this break */
    ushort      spaceCount;     /* # of glue spaces in interword spaces */
    ushort      trailingSpaces; /* # of trailing spaces */
    ushort      chCount;        /* # of chars */
    ushort      fillSpCount;    /* # of fillspaces we have to patch */
    int         lineVal;        /* offset into leadvals of lead */
    eBreakType  breakVal;       /* goodness of break val */
    MicroPoint  minGlue;        /* minimum glue */
    MicroPoint  optGlue;        /* width of optGlue to this point */
    MicroPoint  maxGlue;        /* max glue */
    MicroPoint  curBox;         /* width of immovable to this point */
    MicroPoint  fHangable;      /* width of hanging character if any */
    CharRecordP theChRec;       /* pointer into stream */
    short       specChanged;    /* spec changed since last candidate */
    TypeSpec    spec;           /* spec at this break point */
    scSpecRecord *specRec;

                CandBreak();
    void        Init();

    CandBreak&  operator=( const CandBreak& );
};

class scMaxLineVals {
public:
    Translate( 0, -RLU_BASEfmTop );
}

/* ====================================================================== */

void    scRLURect::RLURomanBaseLineToMiddle( void )
{
    Translate( 0, -RLU_BASEfmTop/2 );
}

/* ====================================================================== */

void    scRLURect::RLURomanBaseLineToBottom( void )
{
    Translate( 0, RLU_BASEfmBottom );
}

/* ====================================================================== */
#if 0
void RectTest( )
{

    scXRect xrect( 100, 100, 200, 200 );
    scMuPoint   pt1;

    scMuPoint   pt2;

    pt1.x = 20;
    pt1.y = 80;

    pt2 = pt1;

    xrect.FourthToThird( 1000 );
    xrect.ThirdToFourth( 1000 );

    pt1.FourthToThird( 1000 );
    pt1.ThirdToFourth( 1000 );


    pt1.FourthToThird( 100 );
    pt1.ThirdToFourth( 100 );


    pt1.FourthToThird( 200 );
    pt1.ThirdToFourth( 200 );


    pt1.FourthToThird( 500 );
    pt1.ThirdToFourth( 500 );



    scAssert( pt1 == pt2 );
}
#endif

/* ====================================================================== */
```

```
void scRLURect::Invalidate( )
{
    Set( SHRT_MAX, SHRT_MAX, SHRT_MIN, SHRT_MIN );
}

/* ==================================================================== */

void scRLURect::Translate( RLU h, RLU v )
{
    rluRight    = rluRight  + h;
    rluLeft     = rluLeft   + h;
    rluTop      = rluTop    + v;
    rluBottom   = rluBottom + v;
}

/* ==================================================================== */

void    scRLURect::FirstToFourth( RLU )
{
    rluTop      = -rluTop;
    rluBottom   = -rluBottom;
}

/* ================================================================== */

void scRLURect::FourthToFirst( RLU )
{
    rluTop      = -rluTop;
    rluBottom   = -rluBottom;
}

/* ================================================================== */

void    scRLURect::RLURomanBaseLineToCenter( void )
{
    rluRight    = (rluRight - rluLeft)/2;
    rluLeft     = 0 - rluRight;

        //use bottom as temp variable to save height
    rluBottom   = rluTop - rluBottom;
    rluTop      = scBaseRLUsystem - ( rluTop + RLU_BASEfmBottom);
    rluBottom   = rluTop + rluBottom;   //Bottom has character height
}

/* ==================================================================== */

void    scRLURect::RLURomanBaseLineToLeft( void )
{
    rluRight    = (rluRight - rluLeft);
    rluLeft     = 0;

        //use bottom as temp variable to save height
    rluBottom   = rluTop - rluBottom;
    rluTop      = scBaseRLUsystem - ( rluTop + RLU_BASEfmBottom);
    rluBottom   = rluTop + rluBottom;   //Bottom has character height
}

/* ==================================================================== */

void    scRLURect::RLURomanBaseLineToRight( void )
{
    rluLeft     = 0 - (rluRight - rluLeft);
    rluRight    = 0;
    rluBottom   = rluTop - rluBottom;   //use bottom as temp variable to save height
    rluTop      = scBaseRLUsystem - ( rluTop + RLU_BASEfmBottom);
    rluBottom   = rluTop + rluBottom;   //Bottom has character height
}

/* ==================================================================== */

void    scRLURect::RLURomanBaseLineToTop( void )
{
```

```
/* ==================================================================== */

void scXRect::FourthToFirst( MicroPoint d )
{
#if SCDEBUG>2
    scAssert( Valid() );
#endif

    y1 -= d;
    y2 -= d;

#if SCDEBUG>2
    scAssert( Valid() );
#endif
}

/* ==================================================================== */

/* ==================================================================== */
/* ==================================================================== */
/* ====================     CRLURECT      =========================== */
/* ==================================================================== */
/* ==================================================================== */


scRLURect::scRLURect( )
{
        // in an attemp to insure that we can freely convert
        // back and forth between these we do the following text
    scAssert( sizeof( scRLURect ) == sizeof( RLU ) * 4 );

    Invalidate();
}

/* ==================================================================== */

scRLURect::scRLURect( const scRLURect& rlurect )
{
    rluLeft     = rlurect.rluLeft;
    rluTop      = rlurect.rluTop;
    rluRight    = rlurect.rluRight;
    rluBottom   = rlurect.rluBottom;
}

/* ==================================================================== */

void scRLURect::Set( RLU left, RLU top, RLU right, RLU bottom )
{
    rluLeft     = left;
    rluTop      = top;
    rluRight    = right;
    rluBottom   = bottom;
}

/* ==================================================================== */

Bool    scRLURect::Valid( eCoordSystem coordSys ) const
{
    switch ( coordSys ) {
        case eFirstQuad:
            return rluLeft <= rluRight && rluTop >= rluBottom;
        case eSecondQuad:
            return rluLeft >= rluRight && rluTop >= rluBottom;
        case eThirdQuad:
            return rluLeft >= rluRight && rluTop <= rluBottom;
        case eFourthQuad:
            return rluLeft <= rluRight && rluTop <= rluBottom;
    }
    return false;
}

/* ==================================================================== */
```

```
#if SCDEBUG>2
    scAssert( Valid() );
#endif

    pt1.x = x2;
    pt1.y = y1;

    pt1.FourthToThird( w );

    pt2.x = x1;
    pt2.y = y2;

    pt2.FourthToThird( w );

    x1 = pt1.x;
    y1 = pt1.y;

    x2 = pt2.x;
    y2 = pt2.y;

#if SCDEBUG>2
    scAssert( Valid() );
#endif
}

/* ==================================================================== */

void scXRect::ThirdToFourth( MicroPoint w )
{
    scMuPoint   pt1,
                pt2;

#if SCDEBUG>2
    scAssert( Valid() );
#endif

    pt1.x = x1;
    pt1.y = y2;

    pt1.ThirdToFourth( w );

    pt2.x = x2;
    pt2.y = y1;

    pt2.ThirdToFourth( w );

    x1 = pt1.x;
    y1 = pt1.y;

    x2 = pt2.x;
    y2 = pt2.y;

#if SCDEBUG>2
    scAssert( Valid() );
#endif
}

/* ==================================================================== */

void scXRect::FirstToFourth( MicroPoint d )
{
#if SCDEBUG>2
    scAssert( Valid() );
#endif

    y1 = -y1;
    y2 = -y2;

#if SCDEBUG>2
    scAssert( Valid() );
#endif

}
```

```
/* ====================     CEXRECT      ============================== */
/* ==================================================================== */
/* ==================================================================== */


#if SCDEBUG > 1

scChar* scXRect::DebugStr( scChar* buf, int factor ) const
{
#if defined(SCWINDOWS)
    wsprintf( buf, scString( "(%d, %d, %d, %d)" ), x1 / factor, y1 / factor, x2 / factor, y2 / facto
r );
#else
    sprintf( buf, scString( "(%d, %d, %d, %d)" ), x1 / factor, y1 / factor, x2 / factor, y2 / factor
 );
#endif
    return buf;
}

#endif

/* ==================================================================== */

Bool scXRect::Valid( eCoordSystem coordSys ) const
{
    switch ( coordSys ) {
        case eFirstQuad:
            return x1 <= x2 && y1 >= y2;
        case eSecondQuad:
            return x1 >= x2 && y1 >= y2;
        case eThirdQuad:
            return x1 >= x2 && y1 <= y2;
        case eFourthQuad:
            return x1 <= x2 && y1 <= y2;
    }
    return false;
}

/* ==================================================================== */

void scXRect::Scale( TenThousandth factor )
{
#if SCDEBUG>2
    scAssert( Valid() );
#endif

    x1 = scRoundMP( (REAL)x1 * factor / 10000.0 );
    x2 = scRoundMP( (REAL)x2 * factor / 10000.0 );
    y1 = scRoundMP( (REAL)y1 * factor / 10000.0 );
    y2 = scRoundMP( (REAL)y2 * factor / 10000.0 );
}

/* ==================================================================== */

void scXRect::Scale( REAL factor )
{
#if SCDEBUG>2
    scAssert( Valid() );
#endif

    x1 = scRoundMP( (REAL)x1 * factor );
    x2 = scRoundMP( (REAL)x2 * factor );
    y1 = scRoundMP( (REAL)y1 * factor );
    y2 = scRoundMP( (REAL)y2 * factor );
}

/* ==================================================================== */

void scXRect::FourthToThird( MicroPoint w )
{
    scMuPoint   pt1,
                pt2;
```

```
/****************************************************************************

    File:       SCHRECT.C

    $Header: /Projects/Toolbox/ct/SCHRECT.CPP 2     5/30/97 8:45a Wmanis $

    Contains:
        This file duplicates in high res rectangles the
            'Calculations on Rectangles' described in Inside MAC I-174


    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

****************************************************************************/

#include "sctypes.h"


#if defined( _MSC_VER )
    #pragma warning(disable:4244)       // disable - int conversion
#endif

/* ================================================================== */
/* ================================================================== */
/* ===================    CMUPOINT    ============================== */
/* ================================================================== */
/* ================================================================== */

void scMuPoint::FourthToThird( MicroPoint w )
{
    MicroPoint  xPrime,
                yPrime;

    scAssert( x != kInvalMP || y != kInvalMP );

    xPrime  = y;
    yPrime  = w - x;
    x       = xPrime;
    y       = yPrime;
}

/* ================================================================== */


void scMuPoint::ThirdToFourth( MicroPoint w )
{
    MicroPoint  xPrime,
                yPrime;

    scAssert( x != kInvalMP || y != kInvalMP );

    xPrime  = w - y;
    yPrime  = x;
    x       = xPrime;
    y       = yPrime;
}

/* ================================================================== */


/* ================================================================== */
/* ================================================================== */
```

```
                eByteOrder   byteOrder )
{
    uchar buf[4];
    raise_if( (*readFunc)( ctxPtr, buf, 4 ) != 4, scERRfile );
    ulong uval;
    BufGet_long( buf, uval, byteOrder );
    val = (long)uval;
}


/* ------------------------------------------------------------ */
// write a quick long

void WriteLong( long            val,
                APPCtxPtr       ctxPtr,
                IOFuncPtr       writeFunc,
                eByteOrder      byteOrder )
{
    uchar buf[4];
    BufSet_long( buf, val, byteOrder );
    raise_if( (*writeFunc)( ctxPtr, buf, 4 ) != 4, scERRfile );
}


/* ------------------------------------------------------------ */

void ReadBytes( uchar*          buf,
                APPCtxPtr       ctxPtr,
                IOFuncPtr       readFunc,
                long            numbytes )
{
    raise_if( (*readFunc)( ctxPtr, buf, numbytes ) != numbytes, scERRfile );
}

/* ------------------------------------------------------------ */

void WriteBytes( const uchar*   buf,
                 APPCtxPtr      ctxPtr,
                 IOFuncPtr      writeFunc,
                 long           numbytes )
{
    raise_if( (*writeFunc)( ctxPtr, (void*)buf, numbytes ) != numbytes, scERRfile );
}

/* ------------------------------------------------------------ */
```

```
                    long extendedRep[3];
                    CvtFloat64To96( extendedRep, &r );
                    SCmemcpy( rbuf, extendedRep, 12 );
                }
#else
                raise( scERRnotImplemented );
#endif
                break;

            default:
                scAssert( 0 );
                break;
        }

        return rbuf + 12;
    }

/* ------------------------------------------------------------ */

/* write out a REAL to a byte buffer */

uchar*      BufSet_REAL( uchar      rbuf[12],
                         REAL       d,
                         eByteOrder  )
{
    switch ( localByteOrder ) {

        case kIntelOrder:
                // convert intel long double to bytes
            IntelDoubleToBytes( rbuf, d );
            break;

        case kMotorolaOrder:
                // convert motorla long double to bytes
            MotorolaDoubleToBytes( rbuf, d );
            break;

        default:
            scAssert( 0 );
            break;
    }

    return rbuf + 12;
}

/* ------------------------------------------------------------ */
/* read in a REAL from a byte buffer */

const uchar*    BufGet_REAL( const uchar    rbuf[12],
                             REAL&          r,
                             eByteOrder )
{
    switch ( localByteOrder ) {
        case kMotorolaOrder:
            BytesToMotorolaDouble( (uchar *)rbuf, r );
            break;

        case kIntelOrder:
            BytesToIntelDouble( (uchar *)rbuf, r );
            break;
        default:
            break;
    }
    return rbuf + 12;
}

/* ------------------------------------------------------------ */
/* ------------------------------------------------------------ */
// write a quick long

void ReadLong( long&        val,
               APPCtxPtr    ctxPtr,
               IOFuncPtr    readFunc,
/* ------------------------------------------------------------ */

    static const uchar* BytesToMotorolaDouble( const REALBUF    rbuf,
                                               REAL&            r )
    {
        uchar   *ptr = (uchar *)&r;

        switch( sizeof( REAL ) ) {
            case 12:
                SCmemcpy( ptr, rbuf, sizeof( REAL ) );
                break;
            case 10:
                ptr[0] = rbuf[0];
                ptr[1] = rbuf[1];
                ptr[2] = rbuf[4];
                ptr[3] = rbuf[5];
                ptr[4] = rbuf[6];
                ptr[5] = rbuf[7];
                ptr[6] = rbuf[8];
                ptr[7] = rbuf[9];
                ptr[8] = rbuf[10];
                ptr[9] = rbuf[11];
                break;
            case 8:
#if 0
                // Convert extended representation to 64 bit IEEE format
                {
                    long extendedRep[3];
                    SCmemcpy( extendedRep, rbuf, 12 );
                    CvtFloat96To64( r, extendedRep );
                }
#else
                raise( scERRnotImplemented );
#endif
                break;
            default:
                scAssert( 0 );
                break;
        }

        return rbuf + 12;
    }

/* ------------------------------------------------------------ */

    static uchar*   MotorolaDoubleToBytes( REALBUF  rbuf,
                                           REAL     r )
    {
        uchar   *ptr = (uchar *)&r;

        switch( sizeof( REAL ) ) {
            case 12:
                SCmemcpy( rbuf, ptr, sizeof( REAL ) );
                break;
            case 10:
                rbuf[0] = ptr[0];
                rbuf[1] = ptr[1];
                rbuf[2] = ptr[0];
                rbuf[3] = ptr[1];
                rbuf[4] = ptr[2];
                rbuf[5] = ptr[3];
                rbuf[6] = ptr[4];
                rbuf[7] = ptr[5];
                rbuf[8] = ptr[6];
                rbuf[9] = ptr[7];
                rbuf[10] = ptr[8];
                rbuf[11] = ptr[9];
                break;
            case 8:
#if 0
                // Convert 64 bit representation to extended
                {
```

```
    return abuf+4;
}

/* ------------------------------------------------------------ */

    static const uchar* BytesToIntelDouble( const REALBUF   rbuf,
                                            REAL&           r )
    {
        uchar   *ptr = (uchar *)&r;

        switch( sizeof( REAL ) ) {
            case 10:
                ptr[9]  = rbuf[2];
                ptr[8]  = rbuf[3];
                break;
            case 8:
                break;
            default:
                scAssert( 0 );
                break;
        }

        ptr[7]  = rbuf[4];
        ptr[6]  = rbuf[5];
        ptr[5]  = rbuf[6];
        ptr[4]  = rbuf[7];
        ptr[3]  = rbuf[8];
        ptr[2]  = rbuf[9];
        ptr[1]  = rbuf[10];
        ptr[0]  = rbuf[11];

        return rbuf + 12;
    }
/* ------------------------------------------------------------ */

    static uchar*   IntelDoubleToBytes( REALBUF     rbuf,
                                        REAL        r )
    {
        uchar   *ptr = (uchar *)&r;

        switch( sizeof( REAL ) ) {
            case 10:
                rbuf[0] = ptr[9];
                rbuf[1] = ptr[8];
                rbuf[2] = ptr[9];
                rbuf[3] = ptr[8];
                break;

            case 8:
                rbuf[0] = ptr[7];
                rbuf[1] = ptr[6];
                rbuf[2] = ptr[7];
                rbuf[3] = ptr[6];
                break;

            default:
                scAssert( 0 );
                break;
        }

        rbuf[4] = ptr[7];
        rbuf[5] = ptr[6];
        rbuf[6] = ptr[5];
        rbuf[7] = ptr[4];
        rbuf[8] = ptr[3];
        rbuf[9] = ptr[2];
        rbuf[10] = ptr[1];
        rbuf[11] = ptr[0];

        return rbuf + 12;
    }
```

```
                *pbuf = (uchar)SC_I2M_MKWORD((uchar*)&s);
                break;

            case kIntelOrder:
                *pbuf = (uchar)SC_M2I_MKWORD((uchar*)&s);
                break;

            default:
                *(ushort*)pbuf = s;
        }

    }
    else
        *(ushort*)pbuf = s;

    return pbuf + sizeof( ushort );
}

/* ------------------------------------------------------------ */
// read out a short from a byte buffer

const uchar*    BufGet_short( const uchar   abuf[2],
                              ushort&       s,
                              eByteOrder    byteOrder )
{
    if ( localByteOrder != byteOrder )
        s = (ushort)SCPIO_MKWORD(abuf);
    else
        s = *(ushort*)abuf;

    return abuf+2;
}

/* ------------------------------------------------------------ */
// write out a long to a byte buffer

uchar*          BufSet_long( uchar      pbuf[4],
                             ulong      l,
                             eByteOrder desiredByteOrder   )
{
    if ( desiredByteOrder != localByteOrder ) {

        switch ( desiredByteOrder ) {

            case kMotorolaOrder:
                *(ulong*)pbuf = SC_I2M_MKLONG((uchar*)&l);
                break;

            case kIntelOrder:
                *(ulong*)pbuf = SC_M2I_MKLONG((uchar*)&l);
                break;

            default:
                *((ulong*)pbuf) = l;
        }
    }
    else
        *((ulong*)pbuf) = l;

    return pbuf + sizeof( ulong );
}

/* ------------------------------------------------------------ */
/* read out a long from a byte buffer */

const uchar*    BufGet_long( const uchar    abuf[4],
                             ulong&         l,
                             eByteOrder     byteOrder )
{
    if ( localByteOrder != byteOrder )
        l = SCPIO_MKLONG(abuf);
    else
        l = *(ulong*)abuf;
```

```
static short localByteOrder = kMotorolaOrder;

#define SCPIO_MKWORD    SC_I2M_MKWORD
#define SCPIO_MKLONG    SC_I2M_MKLONG

#endif

#ifndef SCPIO_MKWORD
    #error "A Processor architecture needs to be defined"
#endif

/* ------------------------------------------------------------ */
/* ------------------------------------------------------------ */
// code for creating the header

uchar*          BufSet_byteorder( uchar pbuf[] )
{
    SCmemset( pbuf, 0, sizeof( ByteOrderStr ) );

    strcpy( (char*)pbuf, byteOrderStr[localByteOrder] );

    return pbuf + sizeof( ByteOrderStr );
}

/* ------------------------------------------------------------ */
// code for extracting the header

const uchar*    BufGet_byteorder( const uchar   pbuf[],
                                  short*        byteOrder )
{
    if ( !strcmp( (char *)pbuf, byteOrderStr[kMotorolaOrder] ) )
        *byteOrder = kMotorolaOrder;
    else if ( !strcmp( (char *)pbuf, byteOrderStr[kIntelOrder] ) )
        *byteOrder = kIntelOrder;
    else
        *byteOrder = kNoOrder;
    return pbuf + sizeof( ByteOrderStr );
}

/* ------------------------------------------------------------ */

uchar*          BufSet_char( uchar*         dstbuf,
                             const uchar*   srcbuf,
                             size_t         bytes,
                             eByteOrder   )
{
    SCmemmove( dstbuf, srcbuf, bytes );
    return dstbuf + bytes;
}

/* ------------------------------------------------------------ */

const uchar*    BufGet_char( const uchar*   srcbuf,
                             uchar*         dstbuf,
                             size_t         bytes,
                             eByteOrder   )
{
    SCmemmove( dstbuf, srcbuf, bytes );
    return srcbuf + bytes;
}

/* ------------------------------------------------------------ */
// write out a short to a byte buffer

uchar*          BufSet_short( uchar         pbuf[2],
                              ushort        s,
                              eByteOrder    desiredByteOrder   )
{
    if ( desiredByteOrder != localByteOrder ) {

        switch ( desiredByteOrder ) {

            case kMotorolaOrder:
```

```
/*****************************************************************

    File:       pfileio.c

    $Header: /Projects/Toolbox/ct/SCFILEIO.CPP 2     5/30/97 8:45a Wmanis $

    Contains:   Implementation of intdependent byte order code.

    Written by: Coletti

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

*****************************************************************/

#include "scfileio.h"
#include "scexcept.h"
#include "scmem.h"
#include <string.h>

#if 0
#include "cvtfloat.h"
#endif
    // the string that goes into the header
static char *byteOrderStr[] = {
    "",
    "Intel86",
    "Motor68",
    NULL
};

/* ------------------------------------------------------------ */
/* ------------------------------------------------------------ */

    // Intel to Motorola
#define SC_I2M_MKWORD(p)    (((ushort) ((p)[1]) << 8 ) | ( (p)[0] ))

    // Motorola to Intel
#define SC_M2I_MKWORD(p)    (((ushort) ((p)[0]) << 8 ) | ( (p)[1] ))


    // Intel to Motorola
#define SC_I2M_MKLONG(p)    \
    ((long)(ulong)SC_I2M_MKWORD(p) | (((long)(ulong)SC_I2M_MKWORD((p)+2)) << 16))

    // Motorola to Intel
#define SC_M2I_MKLONG(p)    \
    ((long)(ulong)SC_M2I_MKWORD((p)+2) | (((long)(ulong)SC_M2I_MKWORD(p)) << 16))


/* ------------------------------------------------------------ */

#if defined( SCWINDOWS ) && !defined( _X86_ )
#define _X86_   1
#endif

#if defined( _X86_ )

static int localByteOrder = kIntelOrder;

#define SCPIO_MKWORD    SC_M2I_MKWORD
#define SCPIO_MKLONG    SC_M2I_MKLONG

#elif defined( SCMACINTOSH )
```

```
        #else
            #define raise(err)                          throw( scException( (status)err ) )
            #define raise_if(exp, err)                  ((exp) ? (throw( scException( (status)err )),0)
: 0)
        #endif
    #endif

#else

    #define raise(err)                          throw( err )
    #define raise_if(exp, err)                  ((exp) ? (throw( err ), 0 ) : 0)

#endif

/* ======================================================================== */
/* ======================================================================== */
/* ======================================================================== */

#endif /* _H_EXCEPT */
```

09585740.072400

```
/*=====================================================================

    File:       EXCEPT.H

    $Header: /Projects/Toolbox/ct/SCEXCEPT.H 2     5/30/97 8:45a Wmanis $

    Contains:   exception code

    Written by: Sealy

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.


=====================================================================*/

#ifndef _H_EXCEPT
#define _H_EXCEPT

#include "sctypes.h"

#if SCDEBUG > 1
    #undef new
#endif

/* ========================================================================= */

class scException {
public:

#if SCDEBUG > 1
                scException( status     errCode = scSuccess,
                             const char* file = 0,
                             int        line = 0 ) :
                                fFile( file ),
                                fLine( line ),
                                fCode( errCode ){ SCDebugBreak(); }

#else
                scException( status     errCode = scSuccess ) :
                                fCode( errCode ){}
#endif

    status      GetValue(void) const            { return fCode; }

#if SCDEBUG > 1
    const char*     fFile;
    const int       fLine;
#endif

private:
    const status    fCode;
};


#if 0

    #if SCDEBUG > 1
        #define raise(err)                      throw( scException( err, __FILE__, __LINE__ ) )
        #define raise_if(exp, err)              ((exp) ? (throw( scException( err, __FILE__, __LINE_
_ )),0) : 0)
    #else
        #ifndef MSVCBUG_1A
            #define raise( scerr )                      throw( scException( scerr ) )
            #define raise_if(exp, scerr )               ((exp) ? (throw( scException( scerr )),0) :
0)
```

```
/*==================================================================

    File:       chfile.h

    $Header: /Projects/Toolbox/ct/SCDBCSDT.H 2     5/30/97 8:45a Wmanis $

    Contains:   Class for reading DBCS files.

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

==================================================================*/



#ifndef _H_SCDBCSDT
#define _H_SCDBCSDT


#include "sctypes.h"

class scDBCSDetector {
public:

    enum eByteType {
        eFirstByte  = -1,
        eOnlyByte,
        eLastByte,
        eMiddleByte
    };      // id's a byte of a multibyte character

                scDBCSDetector( TypeSpec ts );

    void        setDBCS( TypeSpec ts );
    long        StrLen( const char * ) const;

    eByteType   ByteType( uchar ch ) const
                    {
                        return dbcs_ ? shiftjis_[ch] : eOnlyByte;
                    }

private:
    Bool                dbcs_;

    static eByteType    shiftjis_[];
};

#endif /* _H_SCDBCSDT */
```

```
    va_list args;


    if ( fmt && *fmt ) {
        va_start( args, fmt );
        DbgVPrintf( fmt, args );
        va_end( args );
    }
}

/* ====================================================================== */
/* Asserts */

void AssertFailed( const scChar *exp, const char *file, int line )
{
#ifdef SCMACINTOSH
    SCDebugTrace( 0, scString( "(%s,%ld): assert failed: \"%s\"\n" ), file, line, exp );
#else
    SCDebugTrace( 0, scString( "(%s,%d): assert failed: \"%s\"\n" ), file, line, exp );
#endif

    raise( scERRassertFailed );
//  throw( new scException( scERRassertFailed, file, line ) );
}

/* ====================================================================== */
```

```
/*****************************************************************************

    File:       DEBUG.C

    $Header: /Projects/Toolbox/ct/SCDEBUG.CPP 2     5/30/97 8:45a Wmanis $

    Contains:   Debugging routines for composition toolkit.

    Written by: Sealy

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.


*****************************************************************************/

#if defined( SCWINDOWS )
    #include <windowsx.h>
#else
    #include <stdio.h>
#endif

#include <stdarg.h>

#include "scexcept.h"

// Debugger output/interrupts

/* ======================================================================== */

void DbgVPrintf( const scChar*  fmt,
                 va_list        args )
{
#if defined( SCWINDOWS )

    scChar  buf[256];
    int     len;

    wvsprintf( buf, fmt, args );
    len = scStrlen( buf );

    if ( buf[len - 1] == '\n' ) {
        buf[ len - 1] = 0;
        scStrcat( buf, scString( "\r\n" ) );
    }

    OutputDebugString( buf );

#elif defined( SCMACINTOSH )

    scChar  buf[256];
    vsprintf( buf, fmt, args );
    fputs( buf,  stderr );

#endif

}

/* ======================================================================== */

void SCDebugTrace( int level, const scChar* fmt, ... )
{
    extern int scDebugTrace;

    if ( level > scDebugTrace )
        return;
```

```
#define MANUAL_INST
#define ANSI_CLASS_INST

#ifdef MANUAL_INST

#define DEFINE_TEMPLATES
#include "scparagr.h"
#include "scspcrec.h"
#include "scpubobj.h"
#undef DEFINE_TEMPLATE

#ifdef _WINDOWS
#pragma warning ( disable : 4660 )   // duplicate template definitions
#endif

#ifdef ANSI_CLASS_INST
    template class scSizeableArray< char* >;
    template class scSizeableArrayD< stPara >;
    template class scSizeableArrayD< scKeyRecord >;
    template class scSizeableArray< UCS2 >;
    template class scSizeableArrayD< scSpecLocation >;
    template class scSizeableArrayD< scSpecRecord >;
    template class scSizeableArrayD< RefCountPtr< stSpec > >;
#endif // ANSI_CLASS_INST

#endif // MANUAL_INST
```

```
/*==================================================================

    File:       EXCEPT.C

    $Header: /Projects/Toolbox/ct/SCEXCEPT.CPP 2     5/30/97 8:45a Wmanis $

    Contains:   xxx put contents here xxx

    Written by: Sealy

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

==================================================================*/

#include "scexcept.h"
#include <string.h>
```

09639370.073400

```
    eFirstByte, /* 0xe5 */
    eFirstByte, /* 0xe6 */
    eFirstByte, /* 0xe7 */
    eFirstByte, /* 0xe8 */
    eFirstByte, /* 0xe9 */
    eFirstByte, /* 0xea */
    eFirstByte, /* 0xeb */
    eFirstByte, /* 0xec */
    eFirstByte, /* 0xed */
    eFirstByte, /* 0xee */
    eFirstByte, /* 0xef */
    eFirstByte, /* 0xf0 */
    eFirstByte, /* 0xf1 */
    eFirstByte, /* 0xf2 */
    eFirstByte, /* 0xf3 */
    eFirstByte, /* 0xf4 */
    eFirstByte, /* 0xf5 */
    eFirstByte, /* 0xf6 */
    eFirstByte, /* 0xf7 */
    eFirstByte, /* 0xf8 */
    eFirstByte, /* 0xf9 */
    eFirstByte, /* 0xfa */
    eFirstByte, /* 0xfb */
    eFirstByte, /* 0xfc */
    eOnlyByte, /* 0xfd */
    eOnlyByte, /* 0xfe */
    eOnlyByte /* 0xff */
};
```

```
    eFirstByte, /* 0x9c */
    eFirstByte, /* 0x9d */
    eFirstByte, /* 0x9e */
    eFirstByte, /* 0x9f */
    eOnlyByte, /* 0xa0 */
    eOnlyByte, /* 0xa1 */
    eOnlyByte, /* 0xa2 */
    eOnlyByte, /* 0xa3 */
    eOnlyByte, /* 0xa4 */
    eOnlyByte, /* 0xa5 */
    eOnlyByte, /* 0xa6 */
    eOnlyByte, /* 0xa7 */
    eOnlyByte, /* 0xa8 */
    eOnlyByte, /* 0xa9 */
    eOnlyByte, /* 0xaa */
    eOnlyByte, /* 0xab */
    eOnlyByte, /* 0xac */
    eOnlyByte, /* 0xad */
    eOnlyByte, /* 0xae */
    eOnlyByte, /* 0xaf */
    eOnlyByte, /* 0xb0 */
    eOnlyByte, /* 0xb1 */
    eOnlyByte, /* 0xb2 */
    eOnlyByte, /* 0xb3 */
    eOnlyByte, /* 0xb4 */
    eOnlyByte, /* 0xb5 */
    eOnlyByte, /* 0xb6 */
    eOnlyByte, /* 0xb7 */
    eOnlyByte, /* 0xb8 */
    eOnlyByte, /* 0xb9 */
    eOnlyByte, /* 0xba */
    eOnlyByte, /* 0xbb */
    eOnlyByte, /* 0xbc */
    eOnlyByte, /* 0xbd */
    eOnlyByte, /* 0xbe */
    eOnlyByte, /* 0xbf */
    eOnlyByte, /* 0xc0 */
    eOnlyByte, /* 0xc1 */
    eOnlyByte, /* 0xc2 */
    eOnlyByte, /* 0xc3 */
    eOnlyByte, /* 0xc4 */
    eOnlyByte, /* 0xc5 */
    eOnlyByte, /* 0xc6 */
    eOnlyByte, /* 0xc7 */
    eOnlyByte, /* 0xc8 */
    eOnlyByte, /* 0xc9 */
    eOnlyByte, /* 0xca */
    eOnlyByte, /* 0xcb */
    eOnlyByte, /* 0xcc */
    eOnlyByte, /* 0xcd */
    eOnlyByte, /* 0xce */
    eOnlyByte, /* 0xcf */
    eOnlyByte, /* 0xd0 */
    eOnlyByte, /* 0xd1 */
    eOnlyByte, /* 0xd2 */
    eOnlyByte, /* 0xd3 */
    eOnlyByte, /* 0xd4 */
    eOnlyByte, /* 0xd5 */
    eOnlyByte, /* 0xd6 */
    eOnlyByte, /* 0xd7 */
    eOnlyByte, /* 0xd8 */
    eOnlyByte, /* 0xd9 */
    eOnlyByte, /* 0xda */
    eOnlyByte, /* 0xdb */
    eOnlyByte, /* 0xdc */
    eOnlyByte, /* 0xdd */
    eOnlyByte, /* 0xde */
    eOnlyByte, /* 0xdf */
    eFirstByte, /* 0xe0 */
    eFirstByte, /* 0xe1 */
    eFirstByte, /* 0xe2 */
    eFirstByte, /* 0xe3 */
    eFirstByte, /* 0xe4 */
    eOnlyByte, /* 0x53 */
    eOnlyByte, /* 0x54 */
    eOnlyByte, /* 0x55 */
    eOnlyByte, /* 0x56 */
    eOnlyByte, /* 0x57 */
    eOnlyByte, /* 0x58 */
    eOnlyByte, /* 0x59 */
    eOnlyByte, /* 0x5a */
    eOnlyByte, /* 0x5b */
    eOnlyByte, /* 0x5c */
    eOnlyByte, /* 0x5d */
    eOnlyByte, /* 0x5e */
    eOnlyByte, /* 0x5f */
    eOnlyByte, /* 0x60 */
    eOnlyByte, /* 0x61 */
    eOnlyByte, /* 0x62 */
    eOnlyByte, /* 0x63 */
    eOnlyByte, /* 0x64 */
    eOnlyByte, /* 0x65 */
    eOnlyByte, /* 0x66 */
    eOnlyByte, /* 0x67 */
    eOnlyByte, /* 0x68 */
    eOnlyByte, /* 0x69 */
    eOnlyByte, /* 0x6a */
    eOnlyByte, /* 0x6b */
    eOnlyByte, /* 0x6c */
    eOnlyByte, /* 0x6d */
    eOnlyByte, /* 0x6e */
    eOnlyByte, /* 0x6f */
    eOnlyByte, /* 0x70 */
    eOnlyByte, /* 0x71 */
    eOnlyByte, /* 0x72 */
    eOnlyByte, /* 0x73 */
    eOnlyByte, /* 0x74 */
    eOnlyByte, /* 0x75 */
    eOnlyByte, /* 0x76 */
    eOnlyByte, /* 0x77 */
    eOnlyByte, /* 0x78 */
    eOnlyByte, /* 0x79 */
    eOnlyByte, /* 0x7a */
    eOnlyByte, /* 0x7b */
    eOnlyByte, /* 0x7c */
    eOnlyByte, /* 0x7d */
    eOnlyByte, /* 0x7e */
    eOnlyByte, /* 0x7f */
    eOnlyByte, /* 0x80 */
    eFirstByte, /* 0x81 */
    eFirstByte, /* 0x82 */
    eFirstByte, /* 0x83 */
    eFirstByte, /* 0x84 */
    eFirstByte, /* 0x85 */
    eFirstByte, /* 0x86 */
    eFirstByte, /* 0x87 */
    eFirstByte, /* 0x88 */
    eFirstByte, /* 0x89 */
    eFirstByte, /* 0x8a */
    eFirstByte, /* 0x8b */
    eFirstByte, /* 0x8c */
    eFirstByte, /* 0x8d */
    eFirstByte, /* 0x8e */
    eFirstByte, /* 0x8f */
    eFirstByte, /* 0x90 */
    eFirstByte, /* 0x91 */
    eFirstByte, /* 0x92 */
    eFirstByte, /* 0x93 */
    eFirstByte, /* 0x94 */
    eFirstByte, /* 0x95 */
    eFirstByte, /* 0x96 */
    eFirstByte, /* 0x97 */
    eFirstByte, /* 0x98 */
    eFirstByte, /* 0x99 */
    eFirstByte, /* 0x9a */
    eFirstByte, /* 0x9b */
    eOnlyByte, /* 0x0a */
    eOnlyByte, /* 0x0b */
    eOnlyByte, /* 0x0c */
    eOnlyByte, /* 0x0d */
    eOnlyByte, /* 0x0e */
    eOnlyByte, /* 0x0f */
    eOnlyByte, /* 0x10 */
    eOnlyByte, /* 0x11 */
    eOnlyByte, /* 0x12 */
    eOnlyByte, /* 0x13 */
    eOnlyByte, /* 0x14 */
    eOnlyByte, /* 0x15 */
    eOnlyByte, /* 0x16 */
    eOnlyByte, /* 0x17 */
    eOnlyByte, /* 0x18 */
    eOnlyByte, /* 0x19 */
    eOnlyByte, /* 0x1a */
    eOnlyByte, /* 0x1b */
    eOnlyByte, /* 0x1c */
    eOnlyByte, /* 0x1d */
    eOnlyByte, /* 0x1e */
    eOnlyByte, /* 0x1f */
    eOnlyByte, /* 0x20 */
    eOnlyByte, /* 0x21 */
    eOnlyByte, /* 0x22 */
    eOnlyByte, /* 0x23 */
    eOnlyByte, /* 0x24 */
    eOnlyByte, /* 0x25 */
    eOnlyByte, /* 0x26 */
    eOnlyByte, /* 0x27 */
    eOnlyByte, /* 0x28 */
    eOnlyByte, /* 0x29 */
    eOnlyByte, /* 0x2a */
    eOnlyByte, /* 0x2b */
    eOnlyByte, /* 0x2c */
    eOnlyByte, /* 0x2d */
    eOnlyByte, /* 0x2e */
    eOnlyByte, /* 0x2f */
    eOnlyByte, /* 0x30 */
    eOnlyByte, /* 0x31 */
    eOnlyByte, /* 0x32 */
    eOnlyByte, /* 0x33 */
    eOnlyByte, /* 0x34 */
    eOnlyByte, /* 0x35 */
    eOnlyByte, /* 0x36 */
    eOnlyByte, /* 0x37 */
    eOnlyByte, /* 0x38 */
    eOnlyByte, /* 0x39 */
    eOnlyByte, /* 0x3a */
    eOnlyByte, /* 0x3b */
    eOnlyByte, /* 0x3c */
    eOnlyByte, /* 0x3d */
    eOnlyByte, /* 0x3e */
    eOnlyByte, /* 0x3f */
    eOnlyByte, /* 0x40 */
    eOnlyByte, /* 0x41 */
    eOnlyByte, /* 0x42 */
    eOnlyByte, /* 0x43 */
    eOnlyByte, /* 0x44 */
    eOnlyByte, /* 0x45 */
    eOnlyByte, /* 0x46 */
    eOnlyByte, /* 0x47 */
    eOnlyByte, /* 0x48 */
    eOnlyByte, /* 0x49 */
    eOnlyByte, /* 0x4a */
    eOnlyByte, /* 0x4b */
    eOnlyByte, /* 0x4c */
    eOnlyByte, /* 0x4d */
    eOnlyByte, /* 0x4e */
    eOnlyByte, /* 0x4f */
    eOnlyByte, /* 0x50 */
    eOnlyByte, /* 0x51 */
    eOnlyByte, /* 0x52 */
```

```
/*=================================================================

    File:       charbyte.c

    $Header: /Projects/Toolbox/ct/SCDBCSDT.CPP 2     5/30/97 8:45a Wmanis $

    Contains:   DBCS code.

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

=================================================================*/


#include "scdbcsdt.h"
#include "scstcach.h"

/* ================================================================== */
/* ================================================================== */

scDBCSDetector::scDBCSDetector( TypeSpec ts )
    : dbcs_(0)
{
    setDBCS( ts );
}

/* ================================================================== */

void scDBCSDetector::setDBCS( TypeSpec ts )
{
    dbcs_ = ts.ptr() ? scCachedStyle::GetCachedStyle( ts ).GetBreakLang() : false;
}

/* ================================================================== */

long scDBCSDetector::StrLen( const char* str ) const
{
    long len = 0;

    for ( ; *str; ) {
        switch( ByteType ( *str++ ) ) {
            case eOnlyByte:
            case eLastByte:
                len++;
                break;
        }
    }
    return len;
}

/* ================================================================== */


scDBCSDetector::eByteType scDBCSDetector::shiftjis_[] = {
    eOnlyByte, /* 0x00 */
    eOnlyByte, /* 0x01 */
    eOnlyByte, /* 0x02 */
    eOnlyByte, /* 0x03 */
    eOnlyByte, /* 0x04 */
    eOnlyByte, /* 0x05 */
    eOnlyByte, /* 0x06 */
    eOnlyByte, /* 0x07 */
    eOnlyByte, /* 0x08 */
    eOnlyByte, /* 0x09 */
```

```
/***************************************************************************

    File:       SCCTYPE.H

    $Header: /Projects/Toolbox/ct/SCCTYPE.H 2     5/30/97 8:45a Wmanis $

    Contains:   Character types.

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

***************************************************************************/

#ifndef _H_SCCTYPE
#define _H_SCCTYPE

#ifndef _H_SCTYPES
#include "sctypes.h"
#endif

#define sc_ASCII            0x0001
#define sc_SPACE            0x0002
#define sc_PUNC             0x0004
#define sc_DIGIT            0x0008
#define sc_ALPHA            0x0010
#define sc_ACCENT           0x0020
#define sc_SYMBOL           0x0040
#define sc_LOCASE           0x0080
#define sc_UPCASE           0x0100
#define sc_LIGATR           0x0200


extern unsigned short sc_CharType[];

/* for now we assume everything above 255 is alpha, with release of kanji and other
 * versions this will change
 */
#define CTIsAlpha(ch)       ( (ch) < 256 ? ( sc_CharType[(ch)+1] & (sc_ALPHA) ) : true )
#define CTIsSelectable(ch)  ( (ch) < 256 ? ( sc_CharType[(ch)+1] & (sc_ALPHA|sc_DIGIT) ) : true )
#define CTIsDigit(ch)       ( (ch) < 256 ? ( sc_CharType[(ch)+1] & (sc_DIGIT) ) : false )
#define CTIsPunc(ch)        ( (ch) < 256 ? ( sc_CharType[(ch)+1] & (sc_PUNC) ) : false )
#define CTIsUpperCase(ch)   ( (ch) < 256 ? ( sc_CharType[(ch)+1] & (sc_UPCASE) ) : false )
#define CTIsLowerCase(ch)   ( (ch) < 256 ? ( sc_CharType[(ch)+1] & (sc_LOCASE) ) : false )
#define CTIsSpace(ch)       ( (ch) < 256 ? ( sc_CharType[(ch)+1] & (sc_SPACE) ) : false )
#define CTIsSymbol(ch)      ( (ch) < 256 ? ( sc_CharType[(ch)+1] & (sc_SYMBOL) ) : false )
#define CTIsVisible(ch)     ( (ch) < 256 ? ( sc_CharType[(ch)+1] & (sc_ALPHA|sc_DIGIT|sc_ACCENT|sc_P
UNC|sc_SYMBOL) ) : true )

#define CTIsDropCapable(ch) ( CTIsVisible( ch ) && !CTIsSpace( ch ) )

Bool        CTStoreAll( UCS2 );
Bool        CTIsFracBar( UCS2 );
UCS2        CTToLower( UCS2 );
UCS2        CTToUpper( UCS2 );
UCS2        CTToggleCase( UCS2 );

#endif /* _H_SCCTYPE */
```

```
            if ( str )
                col->GetStream()->STRMark( scRETABULATE | scREBREAK );
        }
    }
    scCachedStyle::SetFlowdir( flowDir );
    GetFlowset()->LimitDamage( 0, scReformatTimeSlice );
}

/* ==================================================================== */

void scFlowDir::SetFlow( eCommonFlow cf )
{
    if ( cf == eNoFlow ) {
        linedir_ = eInvalidFlow;
        glyphdir_ = eInvalidFlow;
    }
    else if ( cf == eRomanFlow ) {
        linedir_ = eTopToBottom;
        glyphdir_ = eLeftToRight;
    }
    else if ( cf == eVertJapanFlow ) {
        linedir_ = eRightToLeft;
        glyphdir_ = eTopToBottom;
    }
    else if ( cf == eBidiFlow ) {
        linedir_ = eTopToBottom;
        glyphdir_ = eRightToLeft;
    }
}

/* ==================================================================== */

eCommonFlow scFlowDir::GetFlow() const
{
    if ( linedir_ == eTopToBottom && glyphdir_ == eLeftToRight )
        return eRomanFlow;
    else if ( linedir_ == eRightToLeft && glyphdir_ == eTopToBottom )
        return eVertJapanFlow;
    else if ( linedir_ == eTopToBottom && glyphdir_ == eRightToLeft )
        return eBidiFlow;
    return eNoFlow;
}

/* ==================================================================== */

#if SCDEBUG > 1

void scColumn::DbgPrintInfo( int debugLevel  ) const
{
    SCDebugTrace( debugLevel, scString( "\nSCCOLUMN 0x%08x - firstline 0x%08x\n" ), this, fFirstline
 );

    scTextline* txl;

    for ( txl = fFirstline; txl; txl = txl->GetNext() )
        txl->DbgPrintInfo( debugLevel );
}

/* ==================================================================== */

#endif
```

```
    }
}

/* ================================================================== */

void scColumn::Unlink( scRedispList* redispList )
{
    scColumn*   firstCol;
    scXRect     lineDamage;

        // mark the paras in the container beings unlinked to be rebroken,
        // since they are losing their home, they definately need to
        // be rebroken
        //
    firstCol = GetPrev();
    if ( firstCol == NULL )
        firstCol = GetNext();

    if ( firstCol ) {
        MarkParas();
        FreeLines( true, lineDamage );  /* deletes lines */

        if ( redispList )
            redispList->AddColumn( this, lineDamage );

        scTBObj::Unlink( );
        SetFlowsetStream( 0 );

        firstCol->Renumber( );
        firstCol->Mark( scINVALID );
        firstCol->LimitDamage( redispList, scReformatTimeSlice );
    }
}

/* ================================================================== */

void scColumn::BreakChain( scColumn* col2 )
{
    raise_if( GetNext() != col2, scERRstructure );

    if ( GetStream() )
        GetStream()->STRDeformat();     // remove any layout information

        // break the link
    SetNext( 0 );
    col2->SetPrev( 0 );

    col2->SetFlowsetStream( 0 );    // set the stream in col 2 to nothing
}

/* ================================================================== */

void scColumn::InvertExtents( HiliteFuncPtr func,
                              APPDrwCtx     mat )
{
    scTextline* txl;

    for ( txl = GetFirstline( ); txl; txl = txl->GetNext() )
        txl->InvertExtents( func, mat );
}

/* ================================================================== */
/* set the flow direction of the container */

void scColumn::FlowsetSetFlowdir( const scFlowDir& flowDir )
{
    scColumn*   col = GetFlowset();

    for ( ; col != 0; col = col->GetNext() ) {
        if ( col->GetFlowdir() != flowDir ) {
            col->SetFlowdir( flowDir );
            col->Mark( scINVALID );
            scStream* str = col->GetStream();
    /* renumber */
    for ( ; col1H; col1H = col1H->GetNext() ) {
        if ( col1H == col2H )
            return true;
    }
    return false;
}

/* ================================================================== */

void scColumn::Link( scColumn*      col2,
                     Bool           reformat,
                     scRedispList*  redispList )
{
    scSelection*    select2 = 0;

        // make sure the existing links make sense
    raise_if ( col2->GetPrev(), scERRstructure );

    raise_if( COLLinkSetContains( this, col2 ), scERRstructure );

        /* mark the paras in each to be rebroken */
    MarkParas( );       /* maybe we should only mark the last one */
    col2->MarkParas( );

    col2->FlowsetSetFlowdir( GetFlowdir() );

    if ( FlowsetGetSelection() && !col2->GetSelection() )
        ;       // we are cool
    else if ( !FlowsetGetSelection() && col2->GetSelection() ) {
                // transfer selection
        select2 = col2->FlowsetGetSelection();
        col2->FlowsetRemoveSelection();
        FlowsetSetSelection( select2 );
    }
    else {
        select2 = col2->FlowsetGetSelection();
        col2->FlowsetRemoveSelection();
        delete select2, select2 = 0;
    }

        // do the actual link
    scTBObj::Link( col2 );

        /* patch the stream(s)
         * if either column has a stream we can deal with it easily,
         * if both have it, append stream2 to stream1
         */
    if ( GetStream() && !col2->GetStream() ) {
            /* col1 has a stream */
        SetFlowsetStream( GetStream() );
    }
    else if ( col2->GetStream() && !GetStream() ) {
            /* col2 has a stream */
        SetFlowsetStream( col2->GetStream() );
    }
    else if ( GetStream() && col2->GetStream() ) {
            // both contain streams
        GetStream()->Append( col2->GetStream() );
        SetFlowsetStream( GetStream() );
    }
    else
        /* no column has a stream */;

        // renumber the streams
    Renumber();

        // patch selection

    if ( reformat ) {
        Mark( scINVALID );
        LimitDamage( redispList, scReformatTimeSlice );
```

```
                    // over what is correct
                size.SetWidth( xrect.x2 );
                size.SetDepth( xrect.y2 );
            }
            break;

        case eVertFlex:
        case eFlexShape:
            size.SetWidth( Width() );
            size.SetDepth( LONG_MAX );
            break;

        default:
        case eHorzFlex:
        case eNoShape:
            size    = GetSize();
            break;
    }
}

/* ================================================================== */
/* Determine maximum depth of text from top (or from right in vertical) */

void scColumn::QueryTextDepth( MicroPoint& depth ) const
{
    switch ( GetShapeType()  ) {
        case eVertShape:
            depth = POLYMaxDepth( fVertH );
            break;

        case eRgnShape:
            depth = RGNMaxDepth( fRgnH );
            break;

        case eVertFlex:
            if ( GetFlowdir().IsVertical() ) {
                depth = TextDepth();
                break;
            }
        case eFlexShape:
            depth = LONG_MAX;
            break;

        case eHorzFlex:
            if ( GetFlowdir().IsVertical() ) {
                depth = LONG_MAX;
                break;
            }
        default:
        case eNoShape:
            depth = TextDepth();
            break;
    }
}

/* ================================================================== */

MicroPoint scColumn::TextDepth( ) const
{
    return GetFlowdir().IsHorizontal() ? Depth() : Width();
}

/* ================================================================== */

static Bool COLLinkSetContains( scColumn * col1H,
                                scColumn * col2H )
{
    scColumn *  prevColH;

    /* backup */
    for ( ; col1H && (prevColH = col1H->GetPrev()) != NULL;
            col1H = prevColH )
        ;
```

```
                break;
            }
        }

    switch ( GetShapeType() ) {
        case eFlexShape:
        case eVertFlex:

            if ( GetShapeType() == eFlexShape )
                margins.Set( 0, 0, 0, 0 );
            else
                margins.Set( 0, 0, Width(), 0 );

                    /* add each line to the current extents */
            for ( txl = GetFirstline( ); txl; txl = nextTxl ) {

                if ( GetShapeType() == eFlexShape )
                    margins.x2 = MAX( txl->GetOrigin().y + txl->GetLength(), margins.x2 );

                nextTxl = LNNext( txl );
                if ( !nextTxl ) {        /* last line */

                    margins.y2 = MAX( txl->GetOrigin().y, margins.y2 );

                        /* this makes vertical flex columns the size
                         * of the text baseline plus whatever amount
                         * of text the application wants to add to the bottom
                         */
                    MicroPoint maxlead = txl->MaxLead( spec );
                    if (spec.ptr()) {
                        margins.y2 += CSlastLinePosition( GetAPPName(), spec );
                    }

                    margins.y2 += txl->GetVJOffset();
                }
            }
            break;

        case eHorzFlex:

            margins.Set( 0, 0, 0, Depth() );

            for ( txl = GetFirstline( ); txl; txl = txl->GetNext( ) )
                margins.x2 = MAX( txl->GetOrigin().x + txl->GetLength(), margins.x2 );
            break;

        case eVertShape:
        case eRgnShape:
        case eNoShape:
            margins.Set( 0, 0, Width(), Depth() );
            break;
    }
}

/* ================================================================== */
/* determine maximum possible depth of the column in its local coordinates */

void scColumn::QuerySize( scSize& size ) const
{
    switch ( GetShapeType() ) {

#ifdef ColumnPolygon
        case eVertShape:
            size.SetDept( POLYMaxDepth( fVertH ) );
            break;
#endif /* ColumnPolygon */

        case eRgnShape:
            {
                scXRect xrect;

                RGNGetExtents( fRgnH, xrect );
                    // this is open to some discussion
                for ( txl = col->GetFirstline(); txl; txl = LNNext( txl ) ) {
                    xrect2.Set( txl->GetOrigin().x,
                                txl->GetOrigin().y,
                                txl->GetOrigin().x + CSfirstLinePosition( col->GetAPPName(), tx
l->SpecAtStart( ) ),
                                col->Depth() );
                    margins.Union( xrect2 );
                }

                txl = col->GetLastline( );
                txl->MaxLead( spec );
                xrect2.Set( txl->GetOrigin().x - CSlastLinePosition( col->GetAPPName(), spec ),
                            txl->GetOrigin().y,
                            txl->GetOrigin().x,
                            col->Depth() );

                margins.Union( xrect2 );
            }
            break;

        case eFlexShape:
            txl = col->GetFirstline();
            if ( txl ) {
                margins.Set( txl->GetOrigin().x,
                             txl->GetOrigin().y,
                             txl->GetOrigin().x + CSfirstLinePosition( col->GetAPPName(), txl->S
pecAtStart( ) ),
                             txl->GetMeasure() );

                for ( txl = col->GetFirstline(); txl; txl = LNNext( txl ) ) {
                    xrect2.Set( txl->GetOrigin().x,
                                txl->GetOrigin().y,
                                txl->GetOrigin().x + CSfirstLinePosition( col->GetAPPName(), tx
l->SpecAtStart( ) ),
                                txl->GetMeasure() );
                    margins.Union( xrect2 );
                }

                txl = col->GetLastline( );
                txl->MaxLead( spec );
                xrect2.Set( txl->GetOrigin().x - CSlastLinePosition( col->GetAPPName(), spec ),
                            txl->GetOrigin().y,
                            txl->GetOrigin().x,
                            txl->GetMeasure() );

                margins.Union( xrect2 );
            }
            break;

        case eVertFlex:
            margins.Set( 0, 0, col->Width(), 0 );

            for ( txl = col->GetFirstline(); txl; txl = txl->GetNext() )
                margins.y2 = MAX( txl->GetOrigin().y + txl->GetLength(), margins.y2 );
            break;
    }
}

void scColumn::QueryMargins( scXRect& margins ) const
{
    scTextline  *txl;
    scTextline  *nextTxl;
    TypeSpec    spec;

    if ( GetFlowdir().IsVertical() ) {
        switch ( GetShapeType() ) {
            case eHorzFlex:
            case eVertFlex:
            case eFlexShape:
                COLQueryMarginsVertical( this, margins, GetShapeType() );
                return;
            default:
```

```
    exSize = sizeof(scColumn);

    if ( !GetPrev() ) {
        for ( para = GetStream(); para; para = para->GetNext( ) )
            exSize += para->ExternalSize();
    }
    switch ( GetShapeType() ) {

        case eVertShape:
#ifdef ColumnPolygon
            exSize += POLYExternalSize( fVertH, fShapePieces );
#endif /* ColumnPolygon */
            break;

        case eRgnShape:
            exSize += RGNExternalSize( fRgnH, fShapePieces );
            break;
    }
    exSize += sizeof( scTBObj );        /* NULL OBJECT */
}

/* ================================================================== */

void scColumn::ZeroEnumeration(  )
{
    ZeroEnum();

    if ( !GetPrev() )
        GetStream()->STRZeroEnumeration();

/* ================================================================== */
/* determine extents of the column in its local coordinates */

void scColumn::ComputeInkExtents( )

    scXRect     lineExtents;
    scTextline* txl;

    /* clear rect */
    fInkExtents.Set( 0, 0, 0, 0 );
    /* add each line to the current extents */
    for ( txl = fFirstline; txl; txl = LNNext( txl ) ) {
        txl->QueryExtents( lineExtents, 1 );
        if ( lineExtents.Width() <= 0 )
            lineExtents.x2 = lineExtents.x1 + 1;
        fInkExtents.Union( lineExtents );
    }
}

/* ================================================================== */
/* determine extents of the column in its local coordinates */

    static void COLQueryMarginsVertical( const scColumn*    col,
                                         scXRect&           margins,
                                         int                shapeType )

    {
        scTextline  *txl;
        scMuPoint   translate;
        TypeSpec    spec;
        scXRect     xrect2;

        switch ( shapeType ) {
            case eHorzFlex:
                txl = col->GetFirstline();

                if ( txl ) {
                    margins.Set( txl->GetOrigin().x,
                                 txl->GetOrigin().y,
                                 txl->GetOrigin().x + CSfirstLinePosition( col->GetAPPName(), txl->S
 pecAtStart( ) ),
                                 col->Depth() );
        case eNoShape:
            SetSize( width, depth );
            break;
        case eVertFlex:
            SetWidth( width );
            break;
        case eHorzFlex:
            SetDepth( depth );
            break;
        case eFlexShape:
            SetSize( width, depth );
            break;
    }
    Mark( scINVALID );
    LimitDamage( redispList, scReformatTimeSlice );
}

/* ================================================================= */

void scColumn::Resize( const scSize&    newSize,
                       scRedispList*    redispList )
{
    switch ( GetShapeType() ) {
        case eRgnShape:
        case eVertShape:
            SetSize( newSize );
            return;
        case eNoShape:
            SetSize( newSize );
            break;
        case eVertFlex:
            SetWidth( newSize.Depth() );
            break;
        case eHorzFlex:
            SetDepth( newSize.Width() );
            break;
        case eFlexShape:
            SetSize( newSize );
            break;
    }
    Mark( scINVALID );
    LimitDamage( redispList, scReformatTimeSlice );
}

/* ================================================================= */
/* ENUMERATE THE COLUMN AND ITS STRUCTURES */


void scColumn::Enumerate( long& objEnumerate )
{
    scTBObj::Enumerate( objEnumerate );

        // if the column has no previous members, that is it is
        // the first column of a set of linked columns, enumerate
        // the paragraphs and their text
        //
    if ( !Prev() && fStream )
        fStream->DeepEnumerate( objEnumerate );
}


/* ================================================================= */

/* return the size of this column for storage purposes, the text stream
 * is always stored with the first column, subsequent columns store
 * just the container itself. (this may present problems for paging of text
 * in multipage documents)
 */

void scColumn::ExternalSize( long& exSize )
{
    scContUnit* para;
```

```
        for ( ; txl; txl = nextTxl ) {
            nextTxl = txl->GetNext();
            txl->Delete( colReformatData.fLineDamage );
        }
    }
}

/* ==================================================================== */

void scColumn::SetFlowsetStream( scStream* cu )
{
    scColumn* col;

    for ( col = (scColumn*)FirstInChain(); col; col = col->GetNext() )
        col->SetStream( cu );
}

/* ==================================================================== */
/* free the stream from the column chain */

void scColumn::FreeStream()
{
    if ( fStream ) {
        fStream->STRFree( );
        SetFlowsetStream( 0 );
    }
}

/* ==================================================================== */
/* force the rebreaking of this column */

void scColumn::Rebreak( scRedispList* redispList )
{
        // save the recomposition state
    Bool    saveRecomposeFlag   = GetRecomposition();

    SetRecomposition( true );

    Rebreak2( redispList );

        // restore the saved value
    SetRecomposition( saveRecomposeFlag );

    Unmark( scINVALID );
}

/* ==================================================================== */
/* rebreak of this column */

void scColumn::Rebreak2( scRedispList* redispList )
{
    Mark( scINVALID );

    if ( DamOpen( ) )
        LimitDamage( redispList, scReformatTimeSlice );
}

/* ==================================================================== */
/* give the column a new width & depth, rebreak and return damaged areas */
/* the column measure and/or depth has changed respond accordingly
 * OBVIOUS OPTIMIZATIONS
 *    if depth increases just add stuff
 */

void scColumn::Resize( MicroPoint       width,
                       MicroPoint       depth,
                       scRedispList*    redispList )
{
    switch ( GetShapeType() ) {
        case eRgnShape:
        case eVertShape:
            SetSize( width, depth );
            return;
        return fStream ? fStream->First() : 0;


    scColumn*   prev    = GetPrev();
    scTextline* txl     = 0;
    scContUnit* p       = 0;

        // get last valid line in prev para, presumably
        // the container has been reformatted
    do {
        txl  = prev->GetLastline();
        if ( !txl )
            prev = prev->GetPrev();
    } while ( prev && !txl );

        // get the paragraph of the last line, check to
        // see if the end of the line represents the end
        // of the paragraph, if it does go to the next para
    if ( txl ) {
        p = txl->GetPara();

        if ( txl->GetEndOffset() == p->GetContentSize() )
            p = p->GetNext();
    }
    else
        p = fStream ? fStream->First() : 0;

    return p;
}

/* ===================================================================== */
/* Delete excess lines in the column                                 */

void scColumn::DeleteExcessLines( scContUnit*   para,
                                  scTextline*   lastTxl,
                                  Bool          testGetStrip,
                                  scCOLRefData& colReformatData )
{
    scTextline* txl;
    scTextline* nextTxl;
    Bool        deleteLines     = false;

    if ( lastTxl ) {
        if ( ( txl = LNNext( lastTxl ) ) != NULL )
            deleteLines = true;
    }
    else if ( ( txl = GetFirstline() ) != NULL ) {

        if (  para == NULL || para->GetCount() <= txl->GetPara()->GetCount() ) {
            if ( ! testGetStrip )
                deleteLines = true;
            else {
                scLINERefData   lineData;

                scCachedStyle::SetFlowdir( GetFlowdir() );
                TypeSpec ts = txl->SpecAtStart();
                scCachedStyle::GetCachedStyle( ts );

                lineData.fOrg               = txl->GetOrigin();
                lineData.fMeasure           = txl->GetMeasure();
                lineData.fLogicalExtents    = scCachedStyle::GetCurrentCache().GetLogicalExtents( );
                lineData.fInitialLead.Set( scCachedStyle::GetCurrentCache().GetComputedLead(), scCac
hedStyle::GetCurrentCache().GetFlowdir() );

                if ( !GetStrip( lineData, eStartColBreak, colReformatData ) )
                        /* the first line will not fit, delete them */
                    deleteLines = true;
            }
        }
    }

    if ( deleteLines ) {
        Mark( scREPAINT );      /* if we delete we need to repaint */
```

```
/* ================================================================== */
/* mark all the paras contained within this container to be rebroken */

scContUnit* scColumn::MarkParas( )
{
    scContUnit* firstPara;
    scContUnit* lastPara;
    scContUnit* para;
    scColumn*   contentCol;

    firstPara = FirstPara();

    if ( firstPara ) {
            // in this case the container has some lines
        lastPara = LastPara();
        for ( para = firstPara; para; para = para->GetNext() ) {
            para->Mark( scREBREAK );
            if ( para == lastPara )
                break;
        }
    }
    else {
        /* in this case the container has no lines,
         * we must try to find a neighbor that has
         * some lines, first we look backwards and then
         * we look forwards, we mark what we find and
         * see if they will reformat into the container
         */
        if ( !GetPrev() )
            firstPara = GetStream();
        else {
            contentCol = PrevWithLines();
            if ( contentCol )
                firstPara = contentCol->LastPara();
            else {
                contentCol = NextWithLines( );
                if ( contentCol )
                    firstPara = contentCol->FirstPara();
            }
            if ( !firstPara ) {
                /* this would be executed if no containers had lines
                 * attached to them
                 */
                firstPara = GetStream();
            }
        }
        if ( firstPara )
            firstPara->Mark( scREBREAK );
    }
    return firstPara;
}

/* ================================================================== */
/* return the paragraph of the last line of text in this column */

scContUnit* scColumn::LastPara( ) const
{
    scTextline* txl  = GetLastline( );

    for ( ; txl && txl->Marked( scINVALID ); txl = txl->GetPrev() )
        ;

    return txl ? txl->GetPara() : NULL;
}

/* ================================================================== */
// return the first valid paragraph of this column

scContUnit* scColumn::FirstPara( ) const
{
        // if no previous column the first guy in the stream is
        // the first paragraph
    if ( !GetPrev() )
```

```
                    }

                }
            }
            if ( aColH->GetRecomposition() && aColH->GetStream() )
                STRReformat( aColH, aColH->GetStream(), scReformatTimeSlice, redispList );
            else {
                scSelection* select = aColH->FlowsetGetSelection();
                select->UpdateSelection( );
            }
        }
    }
}

/* ==================================================================== */
/* search a column building a list of typespecs that are contained
 * in the column
 */

void scColumn::GetTSList( scTypeSpecList& tsList ) const

{
    scTextline* txl;

    for ( txl = GetFirstline(); txl; txl = txl->GetNext() )
        txl->GetTSList( tsList );
}

/* ==================================================================== */
/* determine the prev column with a line in it */

scColumn* scColumn::PrevWithLines() const
{
    scColumn* col;

    for ( col = GetPrev(); col ; col = col->GetPrev() ) {
        if ( col->GetFirstline( ) )
            return col;
    }
    return 0;
}

/* ==================================================================== */
/* determine the next column with a line in it */

scColumn* scColumn::NextWithLines( ) const
{
    scColumn* col;

    for ( col = GetNext(); col; col = col->GetNext( ) ) {
        if ( col->GetFirstline( ) )
            return col;
    }
    return 0;
}

/* ==================================================================== */
/* return the last line in this column */

scTextline* scColumn::GetLastline( ) const
{
    scTextline* txl;
    scTextline* validLine = 0;

    for ( txl = fFirstline; txl; txl = txl->GetNext() ) {
        if ( !txl->Marked( scINVALID ) ) {
            validLine = txl;
        //  validLine->AssertValid();
        }
    }
    return validLine;
}
```

```
}

/* ================================================================== */
/* upon completion of reading data in from disk we search down the column list
 * finding the first columns in a chain and retabulate, rebreak and repaint
 */

void scColumn::Update( scRedispList *redispList )
{
    scColumn*   flowset;
    scColumn*   col;

    for ( col = GetBaseContextList(); col; col = col->GetContext() ) {
        if ( col->Marked( scINVALID ) && col->GetRecomposition() ) {
            flowset = (scColumn*)col->FirstInChain();

            scCachedStyle::SetFlowdir( flowset->GetFlowdir() );

            flowset->LimitDamage( redispList, scReformatTimeSlice );

            scColumn* p = flowset;
            for ( ; p; p = p->GetNext() )
                p->Unmark( scINVALID );
        }
    }
}

/* ================================================================== */
/* this is still a little dirty - needs to be cleaned up a bit */
/* reformat all columns containing ts */

void scColumn::ChangedTS( TypeSpec      theTS,
                          eSpecTask     task,
                          scRedispList* redispList )
{
    scColumn*   aColH;
    scContUnit* p;
    scTextline* txl;

    scCachedStyle::StyleInvalidateCache( theTS );

    for ( aColH = GetBaseContextList( ); aColH; aColH = aColH->GetContext() ) {
        if ( aColH->GetCount() == 0 ) {

            scCachedStyle::SetFlowdir( aColH->GetFlowdir() );

            p = aColH->GetStream();
            for ( ; p; p = p->GetNext( ) ) {
                if ( p->ContainTS( theTS ) ) {
                    if ( !aColH->GetRecomposition() ) {
                        if ( ( txl = p->GetFirstline() ) != NULL ) {
                            scColumn * colH;

                            if ( ( colH = txl->GetColumn() ) != NULL )
                                colH->Mark( scINVALID );
                        }
                        else
                            aColH->Mark( scINVALID );

                        if ( task & eSCRetabulate )
                            p->Mark( scRETABULATE );
                        if ( task & eSCRebreak )
                            p->Mark( scREBREAK );
                        if ( task & eSCRepaint )
                            p->ForceRepaint( 0L, LONG_MAX );
                    }
                    else {
                        if ( task & eSCRetabulate )
                            p->Retabulate( theTS );
                        if ( task & eSCRebreak )
                            p->Mark( scREBREAK );
                        if ( task & eSCRepaint )
                            p->ForceRepaint( 0L, LONG_MAX );
```

```
}

/* ================================================================== */
/* draw the portions of the column that intersect the 'damagedRectangle' */

void scColumn::Draw( const scXRect&     dRect,
                     APPDrwCtx          dc,
                     const scMuPoint*   translation )
{
    scTextline* txl;
    scXRect     exRect;
    scMuPoint   tx( 0, 0 );

    if ( translation )
        tx += *translation;

    for ( txl = GetFirstline(); txl != NULL; txl = txl->GetNext() ) {
        txl->QueryExtents( exRect, 1 );
        if ( exRect.Intersect( dRect )  ) {
            txl->Draw( dc, fFlowDir, tx );
            txl->Unmark( scREPAINT );
        }
    }
}

/* ================================================================== */
/* read from a text file */

void scColumn::ReadTextFile( TypeSpec       spec,
                             APPCtxPtr      ctxPtr,
                             IOFuncPtr      readFunc,
                             scRedispList*  redispList )
{
    scColumn*   startCol;

    scCachedStyle::SetFlowdir( GetFlowdir() );
    scCachedStyle::GetCachedStyle( spec );

    startCol = (scColumn*)FirstInChain();

    if ( GetStream() )
        GetStream()->RemoveEmptyTrailingParas( GetFlowset() );

    if ( startCol->GetStream() == NULL )
        SetFlowsetStream( scStream::ReadTextFile( spec, ctxPtr, readFunc, 0 ) );
    else
        startCol->GetStream()->Append( scStream::ReadTextFile( spec, ctxPtr, readFunc, 0 ) );

    startCol->Mark( scINVALID );
    startCol->LimitDamage( redispList, scReformatTimeSlice );      /* reBreak */
}

/* ================================================================== */
/* paste APPText into a text container */

void scColumn::PasteAPPText( stTextImportExport&       appText,
                             scRedispList*  redispList )
{
    scColumn*   firstCol;
    TypeSpec    nullSpec;

    if ( GetStream() )
        GetStream()->RemoveEmptyTrailingParas( GetFlowset() );

    firstCol = GetFlowset();

    if ( !fStream )
        firstCol->SetFlowsetStream( scStream::ReadAPPText( appText ) );
    else
        fStream->Append( scStream::ReadAPPText( appText ) );

    firstCol->Mark( scINVALID );
    firstCol->LimitDamage( redispList, scReformatTimeSlice );       /* reBreak */
```

```
            for ( lineCount = 1;
                    countTxl != NULL;
                        lineCount++, countTxl = LNNext( countTxl ) ) {
                if ( countTxl == txl )
                    return lineCount;
            }
        }
    }
    return -1;
}

/* ==================================================================== */
/* determine the size of the damagerect for ImmediateRedisp depending
 * on the lines set
 */

void scColumn::LineExtents( scImmediateRedisp& immediateRedisp )
{
    scTextline* txl;
    short       count;
    scXRect     colRect;
    scXRect     rect;

    colRect.Invalidate();

    txl = fFirstline;
    for ( count = 1; txl && count < immediateRedisp.fStartLine; count++ )
        txl = txl->GetNext();

    if ( txl ) {
        do {
            txl->QueryExtents( rect );
            colRect.Union( rect );
            txl = txl->GetNext();
            count++;
        } while ( txl && count <= immediateRedisp.fStopLine );
    }

    if ( colRect.Valid() ) {
        if ( fFlowDir.IsHorizontal() ) {
            colRect.x1  = MIN( colRect.x1, 0 );
            colRect.x2  = MAX( colRect.x2, Width() );
        }
        else {
            colRect.y1  = MIN( colRect.y1, 0 );
            colRect.y2  = MAX( colRect.y2, Depth() );
        }
    }

    immediateRedisp.fImmediateRect = colRect;
}

/* ==================================================================== */
/* draw the line of text in the selection */

void scColumn::UpdateLine( scImmediateRedisp&   immediateRedisp,
                           APPDrwCtx            mat )
{
    scTextline* paintTxl;
    short       count;
    scMuPoint   tx( 0, 0 );

    paintTxl = fFirstline;
    for ( count = 1; paintTxl != NULL && count < immediateRedisp.fStartLine; count++ )
        paintTxl = paintTxl->GetNext();

    if ( paintTxl != NULL ) {
        do {
            paintTxl->Draw( mat, GetFlowdir(), tx );
            paintTxl = paintTxl->GetNext();
            count++;
        } while ( paintTxl != NULL && count <= immediateRedisp.fStopLine );
    }
```

```
Bool scColumn::HasText( ) const
{
    scContUnit* p;

    for ( p = GetStream(); p; p = p->GetNext( ) ) {
        if ( p->GetContentSize() > 0 )
            return true;
    }
    return false;
}

/* ==================================================================== */
/* does the text flow out the bottom of this container */

Bool scColumn::MoreText( ) const
{
    scTextline* txl;
    scContUnit* para;
    scColumn*   neighborCol;

    txl = GetLastline();

    if ( txl ) {
        para = txl->GetPara();
        if ( para->GetNext() )
            return true;
        else if ( para->GetContentSize() > txl->GetEndOffset() )
            return true;

    }
    else if ( GetStream() ) {
        neighborCol =  NextWithLines();     // text in subsequent columns
        if ( neighborCol )
            return true;

        neighborCol = PrevWithLines();

        if ( neighborCol ) {
            txl = neighborCol->GetLastline();

                // this gets a little tricky, we are assuming that
                // the text cannot be reformatted into this or
                // some other column and therefore it hangs off the
                // end
            para = txl->GetPara();
            if ( para->GetNext() )
                return true;    // another paragraph beyond last formatted line
            else if ( para->GetContentSize() > txl->GetEndOffset() )
                return true;    // more characters byyond last formatted line
            return false;       // no more text
        }
        return true;            // no text formatted and we have a stream
    }
    return false;
}

/* ==================================================================== */
// determines line num in column of selection, assumes a sliver cursor

short COLLineNum( scSelection* select )
{
    scColumn*   col;
    scTextline* txl;
    scTextline* countTxl;
    short       lineCount;

    if ( select ) {
        col = select->fMark.fCol;
        txl = select->fMark.fTxl;
        if ( col && txl ) {
            countTxl = col->GetFirstline();
    SetFlowsetStream( 0 );
}

/* ==================================================================== */

void scColumn::FlowsetPasteStream( scStream*        stream,
                                   scRedispList*    redispList )
{
    scColumn*   firstCol = GetFlowset();


    stream->STRMark( scREBREAK );

    if ( GetStream() )
        GetStream()->Append( stream );
    else
        SetFlowsetStream( stream );

    firstCol->Mark( scINVALID );
    firstCol->LimitDamage( redispList, scReformatTimeSlice );
}

/* ==================================================================== */
/* free the column, not any text associated with it and unlink it from
 * its column chain
 */

void scColumn::Delete( scRedispList* redispList )
{
    scColumn* firstCol;
    scColumn* nextCol;


    firstCol        = (scColumn*)FirstInChain();
    nextCol         = GetNext();

    if ( this == firstCol ) {
                // trying to free a column in a chain without
                // unllinking it
        raise_if( nextCol && GetStream(), scERRstructure );

        if ( !nextCol && GetStream() ) {
                // we are the only column left so
                // we need to delete the text stream
            FreeStream( );

            TypeSpec nullSpec;

                // clear the cache to help eliminate refs to specs
            scCachedStyle::StyleInvalidateCache( nullSpec );
        }
    }
    scTBObj::Unlink( );
    DeleteFromCTXList( );


    if ( firstCol != this ) {
        firstCol->Renumber( );
        firstCol->Mark( scINVALID );
        firstCol->LimitDamage( redispList, scReformatTimeSlice );
    }
    else if ( nextCol ) {
        firstCol->Renumber( );
        firstCol->Mark( scINVALID );
        firstCol->LimitDamage( redispList, scReformatTimeSlice );
    }

    Free();
}

/* ==================================================================== */
// because of reformatting nothing lands in here we will still return
// true
```

```
/* free the column, no disentanglement of pointers, save its own internal
 * structures
 */

void scColumn::Free()
{
    scXRect lineDamage;

    FreeLines( false, lineDamage );     // deletes lines
    SetShapeType( eNoShape );

        // free it up from the context list
    DeleteFromCTXList( );

    delete this;
}

/* ================================================================== */
/* free the column that is part of the scrap */

void scColumn::FreeScrap( )
{
    scAssert( !GetNext() );

    FreeStream( );    /* deletes stream */

    DeleteFromCTXList( );
    Free();
}

/* ================================================================== */
/* clear the stream from the set of linked columns,
 * that this column belongs to
 */

void scColumn::FlowsetClearStream( scRedispList* redispList )
{
    scColumn*   firstCol = GetFlowset();
    scXRect     lineDamage;

        // invalidate selection
    FlowsetInvalidateSelection();

        // free all the lines associated with the column(s)
    scColumn* col;
    for ( col = firstCol; col; col = col->GetNext() ) {
        if ( col->GetFirstline() )
            col->FreeLines( true, lineDamage );  /* deletes lines */
    }

        // delete the stream from all the column(s)
    firstCol->FreeStream( );
}

/* ================================================================== */
/* cut the stream from the set of linked columns,
 * that this column belongs to
 */

void scColumn::FlowsetCutStream( scStream*      stream,
                                 scRedispList*  redispList )
{
    scColumn*   firstCol = GetFlowset();
    scXRect     lineDamage;

    FlowsetInvalidateSelection();

    stream->STRDeformat();

    scColumn* col;
    for ( col = firstCol; col; col = col->GetNext() )
        col->FreeLines( true, lineDamage );  /* deletes lines */
```

```
                 * into a flexible container - we will have to free the shape
                 */
                FreeShape();
            }
            if ( type == eNoShape )
                fLayBits.fLayType = type;
            else
                fLayBits.fLayType = (eColShapeType)(( fLayBits.fLayType & eFlexShape ) | type );
            break;
    }
}

/* ==================================================================== */
/* free the lines with the column, this is tricky because we may want
 * to disentangle pointers at the same time
 */

void scColumn::FreeLines( Bool      reportDamage,
                          scXRect&  lineDamage )
{
    scTextline* txl;
    scTextline* nextTxl;
    scContUnit* para;
    scXRect     extents;
    scContUnit* streamPresent = fStream;

    for ( txl = fFirstline; txl; txl = nextTxl ) {
#if SCDEBUG > 1
        txl->scTBObj::scAssertValid();
#endif
        nextTxl = LNNext( txl );
        if ( reportDamage ) {
            txl->QueryExtents( extents, 1 );
            if ( extents.Width() == 0 )
                extents.x2 = extents.x1 + 1;
            lineDamage.Union( extents );
        }
        if ( streamPresent != 0 ) {
            para = txl->GetPara( );
            if ( para && para->GetFirstline() == txl )
                para->SetFirstline( 0 );
        }
        delete txl;
    }

    SetFirstline( NULL );
}

/* ==================================================================== */
/* free the vertices of this column */

void scColumn::FreeShape( )
{
    switch ( GetShapeType() ) {
        case eVertShape:
            if ( fVertH != NULL )
                MEMFreeHnd( fVertH );
            fShapePieces = 0;
            fVertH = NULL;
            break;
        case eRgnShape:
            if ( fRgnH != NULL )
                DisposeHRgn( fRgnH );
            fShapePieces = 0;
            fRgnH = NULL;
            break;
        default:
            break;
    }
}

/* ==================================================================== */
```

```
        if ( this == col )
            return;
    }

    raise( scERRidentification );
}

/* ================================================================== */

void scColumn::FiniCTXList( void )
{
    scColumn*   col;
    scColumn*   nextCol;

    for ( col = fTheContextList; col; col = nextCol ) {

        SCDebugTrace( 1, scString( "FiniCTXList: 0x%08x\n" ), col );

        nextCol = col->GetContext();

        col->FreeStream( );

            // must do this since all layout are tracked
        col->scTBObj::Unlink();
        col->Free();
    }
}

/* ================================================================== */

void scColumn::SetVertFlex( Bool            tf,
                            scRedispList*   redispList )
{
    if ( tf )
        SetShapeType( eVertFlex );
    else
        fLayBits.fLayType = (eColShapeType)( fLayBits.fLayType & ~eVertFlex );

    Mark( scINVALID );
    LimitDamage( redispList, scReformatTimeSlice );
}

/* ================================================================== */

void scColumn::SetHorzFlex( Bool            tf,
                            scRedispList*   redispList )
{
    if ( tf )
        SetShapeType( eHorzFlex );
    else
        fLayBits.fLayType = (eColShapeType)( fLayBits.fLayType & ~eHorzFlex );

    Mark( scINVALID );
    LimitDamage( redispList, scReformatTimeSlice );
}

/* ================================================================== */

void scColumn::SetShapeType( eColShapeType type )
{
    switch ( type ) {
        case eVertShape:
        case eRgnShape:
            if ( (eColShapeType)fLayBits.fLayType != type )
                FreeShape();
            fLayBits.fLayType = type;
            break;
        case eVertFlex:
        case eHorzFlex:
        case eFlexShape:
        case eNoShape:
            if ( (eColShapeType)fLayBits.fLayType & eIrregShape ) {
                /* we are trying to turn an irregularly shaped container
        col->SetCount( count++ );
}

/* ================================================================ */
/* creates a new unlinked empty column of specified width and depth */

scColumn::scColumn( APPColumn   appName,
                    MicroPoint  width,
                    MicroPoint  depth,
                    scStream*   p,
                    eCommonFlow flow ) :
                        fShapePieces( 0 ),
                        fRgnH( 0 ),
                        fNextContext( 0 ),
                        fAppName( appName ),
                        fColumnCount( 0 ),
                        fSize( width, depth ),
                        fFlowDir( flow ),
                        fStream( p ),
                        fSelection( 0 ),
                        fFirstline( 0 )
{
    SetShapeType( eNoShape );

    fInkExtents.Set( 0, 0, 0, 0 );

        /* add to context list */
    AddToCTXList( );

    if ( appName == 0 )
        fAppName = (APPColumn)this;
}

/* ================================================================ */

scColumn* scColumn::FindFlowset( const scStream* str )
{
    scColumn *col;

    col          = fTheContextList;

    for ( ; col; col = col->GetContext() ) {
        if ( col->GetStream() == str )
            return col->GetFlowset();
    }
    return 0;
}

/* ================================================================ */

void scColumn::DeleteFromCTXList( )
{
    scColumn *col;

    col          = fTheContextList;

    if ( this == col )
        fTheContextList = GetContext();
    else {
        for ( ; col && col->GetContext() != this; col = col->GetContext() )
            ;
        if ( col )
            col->SetContext( GetContext() );
    }
}

/* ================================================================ */

void scColumn::VerifyCTXList( void ) const
{
    register scColumn* col;

    for ( col = fTheContextList; col; col = col->GetContext() ) {
```

```
    if ( fVertH && GetShapeType() == eVertShape ) {
        scrapPolyH = MEMAllocHnd( fShapePieces * sizeof(scVertex) );

        scAutoUnlock    h1( scrapPolyH );
        scAutoUnlock    h2( fVertH );
        SCmemmove( (scVertex*)*h1, (scVertex*)*h2, (size_t)(fShapePieces * sizeof( scVertex ) ) );
    }
    else
        scrapPolyH = NULL;

    *dstVertHP = scrapPolyH;
}

/* ==================================================================== */

#endif

/* ==================================================================== */
/* this is primarily called when a column has changed, it forces a rebreak
 * of the paragraphs in the column, 'StrRerfomat' should take care of damage
 * to subsequent paragraphs in subsequent columns, this also forces the
 * the rebreaking of any paragraphs that have no first line, thus if
 * a column is deleted it will force the correct rebreaking
 */

void scColumn::LimitDamage( scRedispList* redisplist, long ticks )
{
    scContUnit* firstPara;
    scColumn*   nextcol;

    /* look thru the stream until we find an intersection of a paragraph
     * and a column, once we have an intersection we mark all the remaining
     * paragraphs to be rebroken, one problem is that if the column has been
     * made so small no lines are in it, then no paras are marked, the code
     * following the walk down the list takes care of that case
     */

    if ( !GetRecomposition() ) {
        Mark( scINVALID );
        return;
    }

    if ( !GetStream() )
        return;

    if ( Marked( scINVALID ) )
        firstPara   = MarkParas( );
    else
        firstPara   = GetStream();

    /* before we get into the stream make sure all paras that need to
     * be marked as REBREAK are marked as such
     */
    for ( nextcol = this; nextcol; nextcol = nextcol->GetNext() ) {
        if ( nextcol->Marked( scINVALID ) )
            nextcol->MarkParas();
    }

    scAssert( firstPara != 0 );

    STRReformat( this, firstPara, ticks, redisplist );
}

/* ==================================================================== */
/* renumber all the columns */

void scColumn::Renumber( )
{
    scColumn* col = (scColumn*)FirstInChain();
    long        count;

        /* renumber */
    for ( count = 0; col; col = col->GetNext() )
    Mark( scINVALID );
    LimitDamage( redispList, scReformatTimeSlice );
}

/* ================================================================== */

#if defined( scColumnShape )

/* add a polygon into the indicated column, rebreak and return
 * damaged areas
 */

void scColumn::PastePoly( scVertHandle  srcVertH,
                          scRedispList* redispList )
{
    ushort          shapePieces;
    scVertex        *srcV,
                    *dstV;

    raise_if( GetShapeType() == eRgnShape, scERRstructure );

    scAutoUnlock h( srcVertH );
    srcV = (scVertex*)*h;

    shapePieces = POLYCountVerts( srcV );

    SetShapeType( eVertShape );

    fVertH = MEMResizeHnd( fVertH, shapePieces * sizeof(scVertex) );


    scAutoUnlock h2( fVertH );
    dstV = (scVertex*)*h2;

    if ( fShapePieces ) {
        dstV += ( fShapePieces - 1 );
        scAssert( dstV->fPointType == eFinalPoint );
        dstV->fPointType = eStopPoint;
        dstV++;
    }

    SCmemmove( dstV, srcV, (size_t)(shapePieces * sizeof( scVertex )) );
    fShapePieces = (ushort)(fShapePieces + shapePieces);

    Mark( scINVALID );
    LimitDamage( redispList, scReformatTimeSlice );
}

/* ================================================================== */

void scColumn::ClearShape( scRedispList* redispList )
{
    switch ( GetShapeType() ) {
        case eVertShape:
        case eRgnShape:
            SetShapeType( eNoShape );
            Mark( scINVALID );
            LimitDamage( redispList, scReformatTimeSlice );
            break;
        case eVertFlex:
        case eHorzFlex:
        case eFlexShape:
        case eNoShape:
            break;
    }
}

/* ================================================================== */

void scColumn::CopyPoly( scVertHandle* dstVertHP )
{
    scVertHandle    scrapPolyH;
```

```
}

/* ================================================================== */
/* paste a region into the indicated column, rebreak and return
 * damaged areas
 */

void scColumn::PasteRgn( const HRgnHandle   srcRgnH,
                         scRedispList*      redispList )
{
    HRgnHandle  dstRgnH;
    HRgn*       rgn;

    raise_if( srcRgnH == NULL, scERRstructure );

    dstRgnH = NewHRgn( RGNSliverSize( srcRgnH ) );

    if ( fRgnH ) {
        SectHRgn( fRgnH, srcRgnH, dstRgnH );
        DisposeHRgn( fRgnH );
    }
    else {
        CopyHRgn( dstRgnH, srcRgnH );
        SetShapeType( eRgnShape );
    }

    fRgnH = dstRgnH;
    scAutoUnlock h( fRgnH );
    rgn = (HRgn *)*h;

    fShapePieces = (ushort)rgn->fNumSlivers;



    Mark( scINVALID );
    LimitDamage( redispList, scReformatTimeSlice );
}

/* ================================================================== */

void scColumn::CopyRgn( HRgnHandle& dstRgn )
{
    dstRgn = NewHRgn( RGNSliverSize( fRgnH ) );
    CopyHRgn( dstRgn, fRgnH );
}

/* ================================================================== */
/* paste a polygon into the indicated column, rebreak and return
 * damaged areas
 */

void scColumn::ReplacePoly( scVertHandle    srcVertH,
                            scRedispList*   redispList )
{
    ushort          shapePieces;
    scVertex*       srcV;
    scVertex*       dstV;

    scAutoUnlock    h( srcVertH );
    srcV = (scVertex*)*h;

    shapePieces = POLYCountVerts( srcV );

    fVertH = MEMResizeHnd( fVertH, shapePieces * sizeof( scVertex ) );

    scAutoUnlock h1( fVertH );
    dstV = (scVertex*)*h1;
    SCmemmove( dstV, srcV, (size_t)(shapePieces * sizeof( scVertex )) );
    fShapePieces = shapePieces;

        /* check if poly type is set */
    SetShapeType( eVertShape );
    select->Restore( &mark, 0, 0, false );

    select->fPoint.fCol = this;

    if ( Select( pt, &select->fPoint, &dist ) )
        select->LineHilite( func );
    else
        raise( scERRlogical );
}

/* ================================================================== */

void scColumn::InitialSelection( TypeSpec&      ts,
                                 scSelection*&  select )
{
    scMuPoint   mPt;
    TextMarker  tm;
    REAL        dist;
    scContUnit* firstPara;
    Bool        iAdded  = false;

    select = NULL;

    raise_if( GetPrev(), scERRlogical );

    if ( !GetStream() ) {
        firstPara = scContUnit::Allocate( ts, NULL, 0L );

            // initialize spec cache
        scCachedStyle::SetParaStyle( firstPara, ts );
        scCachedStyle::GetCachedStyle( ts );

        SetFlowsetStream( (scStream*)firstPara );

        Mark( scINVALID );
        LimitDamage( 0, scReformatTimeSlice );
        iAdded = true;
    }

    mPt.Set( 0, 0 );

    if ( !Select( mPt, &tm, &dist ) ) {
        if ( iAdded )
            FreeStream( );
        raise( scERRstructure );
    }

    select = FlowsetGetSelection();
    select->SetMark( tm );
    select->SetPoint( tm );
}

/* ================================================================== */

void scColumn::LineInfo( scLineInfoList*    lineInfoList,
                         long&              nLines,
                         Bool&              moreText ) const
{
    scTextline* txl;

    nLines      = GetLinecount();
    moreText    = MoreText( );

    if ( lineInfoList && nLines ) {
        scLineInfo  lineInfo;

        lineInfoList->RemoveAll();

        for ( txl = GetFirstline(); txl; txl = txl->GetNext() ) {
            txl->GetLineInfo( lineInfo );
            lineInfoList->AppendData( (ElementPtr)&lineInfo );
        }
    }
```

```
void scColumn::StartClick( const scMuPoint& pt,
                           HiliteFuncPtr    func,
                           APPDrwCtx,
                           scSelection*&    select )
{
    REAL        dist;
    scSelection selection;

    if ( !GetStream() )
        return;

    selection.fMark.fCol    = this;

    COLSetSelMax( this, &selection.fMark, pt );

    raise_if ( !Select( pt, &selection.fMark, &dist ), scERRlogical );

    selection.fPoint    = selection.fMark;

    selection.LineHilite( func );

    select = FlowsetGetSelection();

    *select = selection;
}

/* ================================================================== */

void scColumn::ContinueClick( const scMuPoint&  pt,
                              HiliteFuncPtr     func,
                              scSelection*      select )
{
    REAL        dist;
    scSelection oldSelection( *select );

    raise_if( !select->fMark.fCol, scERRstructure );

    if ( !GetStream() )
        return;

    select->fPoint.fCol = this;

    if ( !GetFirstline() )
        return;

        // columns not in same stream, application program should catch this
    raise_if ( select->fMark.fCol->GetStream( ) != select->fPoint.fCol->GetStream(), scERRstructure
);

    COLSetSelMax( this, &select->fPoint, pt );

    if ( Select( pt, &select->fPoint, &dist ) ) {
        select->InteractiveHilite( oldSelection, func );
    }
    else
        raise( scERRlogical );
}

/* ================================================================== */

void scColumn::StartShiftClick( scStreamLocation&   mark,
                                const scMuPoint&    pt,
                                HiliteFuncPtr       func,
                                APPDrwCtx,
                                scSelection*&       select )
{
    REAL    dist;

    if ( !GetStream() )
        return;

    select = FlowsetGetSelection();
```

```
/* ================================================================== */
/* select something special indicated by the SelectType */

void scColumn::SelectSpecial( const scMuPoint&  pt,
                              eSelectModifier   selectMod,
                              scSelection*&     select )
{
    select = FlowsetGetSelection();

    scSelection     newSelection( *select );
    REAL            dist;

    if ( !GetStream() )
        return;


    newSelection.fMark.fCol     = this;
    COLSetSelMax( this, &newSelection.fMark, pt );

    if ( selectMod == eAllSelect )
        newSelection.AllSelect( );
    else {

#ifdef TESTEXTENTS

        {
            HRect           maxExRect,          /* column extents */
                            maxMargRect;        /* column margins */

            /* if the point is to far out of the maxExRect
             * things will get very slow
             */

            maxExRect = col->fExtents;
            SetHRect( &maxMargRect, 0, 0, col->fWidth, col->fDepth );

            UnionHRect( &maxExRect, &maxMargRect, &maxExRect );


            if ( !MuPtInHRect( pt, &maxExRect ) ) {
                /* the point is in GM's front yard */
                return scERRbounds;
            }
        }
#endif /* TESTEXTENTS */


        raise_if( !Select( pt, &newSelection.fMark, &dist ), scERRbounds );

        newSelection.fPoint   = newSelection.fMark;

        switch ( selectMod ) {
            case eWordSelect:
                newSelection.WordSelect();
                break;
            case eLineSelect:
                newSelection.LineSelect();
                break;
            case eParaSelect:
                newSelection.ParaSelect();
                break;
            case eColumnSelect:
                newSelection.ColumnSelect();
        }
    }

    *select = newSelection;
}


/* ================================================================== */
/* start selection in the original column
 */
```

```
                    textMarker->fPara       = txl->GetPara();
                    textMarker->fTxl        = txl;
                    textMarker->fParaCount  = textMarker->fPara->GetCount();
                    textMarker->fLineCount  = txl->GetLinecount();
                }
            }
                /* if no selection and the y position is
                 * lower than the top of the last line, then
                 * select the last char on the last line
                 */

                /* assumes lines move from right to left */
            if ( vertical )
                belowText = !LNNext( txl )  && hitPt.x < exRect.x2 && *bestDist == DBL_MAX;
            else
                belowText = !LNNext( txl )  && hitPt.y > exRect.y2 && *bestDist == DBL_MAX;

            if ( belowText ) {
                *bestDist = txl->Select( charOrg, count, hitPt, eCursForward, textMarker->fEndOfLine
 );

                textMarker->fOffset     = txl->GetEndOffset();

                scMuPoint charOrg;
                charOrg                 = txl->Locate( textMarker->fOffset, charOrg, eCursForward );
                if ( vertical )
                    textMarker->fHLoc   = charOrg.y;
                else
                    textMarker->fHLoc   = charOrg.x;

                textMarker->fPara       = txl->GetPara();
                textMarker->fTxl        = txl;
                textMarker->fParaCount  = textMarker->fPara->GetCount();
                textMarker->fLineCount  = txl->GetLinecount();
                break;
            }
        }

        if ( vertical ) {
            fudgeHFactor -= scPOINTS(1);
            fudgeVFactor -= scPOINTS(8);
        }
        else {
            fudgeHFactor -= scPOINTS(144);
            fudgeVFactor -= scPOINTS(1);
        }
    }

    return( textMarker->fPara != NULL );
}

/* ==================================================================== */
/* return a number that is the square of the dx plus the square of the
 * dy between the 'pt' and a significant point
 */

void scColumn::ClickEvaluate( const scMuPoint&  pt,
                              REAL&             dist )
{
    TextMarker  tm;
    REAL        nearDist;

    dist = DBL_MAX;     /* defined in scmath.h */

    if ( GetStream() ) {
        tm.fCol = this;
        raise_if ( !Select( pt, &tm, &nearDist ), scERRlogical );
    }

    dist = nearDist;
}
```

```
        else
            txl->Hilite( NULL, LONG_MIN, NULL, LONG_MAX, appMat, func, selection );
    }
}

/* ================================================================== */
/* select text in a col at the given hit point */

Bool scColumn::Select( const scMuPoint& hitPt,
                       TextMarker*      textMarker,
                       REAL*            bestDist )
{
    scXRect     exRect;
    scTextline* txl;
    long        count;
    scMuPoint   charOrg;
    MicroPoint  fudgeHFactor,
                fudgeVFactor;
    REAL        dist;
    Bool        belowText   = false;
    Bool        vertical    = false;
    int         lineNum;

    vertical = GetFlowdir().IsVertical();

        /* make first hit infinitely far away */
    *bestDist = DBL_MAX;

    textMarker->fCol        = this;
    textMarker->fColCount   = GetCount();
    textMarker->fPara       = NULL;
    textMarker->fTxl        = NULL;

    fudgeHFactor = fudgeVFactor = 0;

    while ( GetFirstline() && !textMarker->fPara ) {
        for ( lineNum = 0, txl = GetFirstline(); txl; txl = LNNext( txl ), lineNum++ ) {

            txl->QueryExtents( exRect );
                // grow hit by fudge factor to account for sloppy hits,
                // how well will this worked on zoomed text?, this value
                // is in world coordinates, NOT the screen coordinates
            exRect.Inset( fudgeHFactor, fudgeVFactor );

            if ( exRect.PinRect( hitPt ) ) {
#if SCDEBUG > 1
                SCDebugTrace( 2, scString( "COLSelect: line #%d (%d,%d) (%d %d %d %d)\n" ),
                              lineNum,
                              muPoints( hitPt.x ), muPoints( hitPt.y ),
                              muPoints( exRect.x1 ), muPoints( exRect.y1 ),
                              muPoints( exRect.x2 ), muPoints( exRect.y2 ) );
#endif
                Bool endOfLine;

                    // we have a hit within the extents of the line, now see
                    // exactly where on the line we may have selected
                dist = txl->Select( charOrg, count, hitPt, eCursNoMovement, endOfLine );

                if ( dist < *bestDist ) {
                        // we have a hit that is better than any previous hit
                    *bestDist = dist;

                    if ( vertical )
                        textMarker->fHLoc       = charOrg.y;
                    else
                        textMarker->fHLoc       = charOrg.x;

                    textMarker->fOffset     = count;

//                  if ( LNOrigin( txl ) + LNLength(txl) <= fHLoc && LNIsHyphenated( txl ) )
//                      textMarker->fEndOfLine  = true;
//                  else
                        textMarker->fEndOfLine  = endOfLine;
            scXRect lineDamage;
            rdl.LineListChanges( this, lineDamage, redispList );
            col->Unmark( scREALIGN );
        }
    }
    scSelection* select = FlowsetGetSelection();
    select->UpdateSelection( );
}

/* ================================================================== */
/* ================================================================== */
#if SCDEBUG > 1

void scColumn::scAssertValid( Bool recurse ) const
{
    scTBObj::scAssertValid( recurse );
    if ( !recurse ) {
        if ( fFirstline )
            fFirstline->scAssertValid( false );

        if ( fStream )
            fStream->scAssertValid( false );
    }
}

#endif

/* ================================================================== */
// should we reformat this column or wait till later

Bool scColumn::DamOpen( )
{
    return APPRecomposeColumn( GetAPPName() );
}

/* ================================================================== */
/* set the max selection extent based upon the column flow direction */

static void COLSetSelMax( scColumn*         col,
                          TextMarker*       tm,
                          const scMuPoint&  muPt )
{
    if ( col->GetFlowdir().IsVertical() )
        tm->fSelMaxX    = muPt.y;
    else
        tm->fSelMaxX    = muPt.x;
}

/* ================================================================== */
/* hilite or dehilite the characters in this column */

void scColumn::Hilite( const TextMarker&        tmMark,
                       const TextMarker&        tmPoint,
                       HiliteFuncPtr            func,
                       const scSelection&       selection )
{
    scTextline* txl;
    scTextline* lastTxl;
    scTextline* txl1        = tmMark.fTxl;
    scTextline* txl2        = tmPoint.fTxl;
    MicroPoint  startLoc    = tmMark.fHLoc,
                stopLoc     = tmPoint.fHLoc;
    APPDrwCtx   appMat;

    APPDrawContext( GetAPPName(), this, appMat );

    lastTxl = txl2->GetNext();

    for ( txl = txl1; txl && txl != lastTxl; txl = txl->GetNext() ) {
        if ( txl == txl1 )
            txl->Hilite( &tmMark, startLoc, NULL, LONG_MAX, appMat, func, selection );
        else if ( txl == txl2 )
            txl->Hilite( NULL, LONG_MIN, &tmPoint, stopLoc, appMat, func, selection );
```

```
Bool scColumn::GetRecomposition( void ) const
{
    scColumn* col = (scColumn*)FirstInChain();

    return col->Marked( scLAYcomposeACTIVE );
}

/* ================================================================== */
// get the selection object associated with the flowset, if there is
// none it will create one

scSelection* scColumn::FlowsetGetSelection( void )
{
    scColumn*       col     = (scColumn*)FirstInChain();

    if ( !col->GetSelection() )
        col->SetSelection( SCNEW scSelection( col ) );

    return col->GetSelection();
}


/* ================================================================== */
// set the selection object for the flowset none should exist,
// since if it does error recovery might be a bit tricky

void scColumn::FlowsetSetSelection( scSelection* sel )
{
    scColumn*   col = (scColumn*)FirstInChain();

    col->SetSelection( sel );
}

/* ================================================================== */
// this removes the selection from the flowset
// NOTE: it does not delete it

void scColumn::FlowsetRemoveSelection( void )
{
    scColumn*   col = (scColumn*)FirstInChain();

    col->SetSelection( 0 );
}

/* ================================================================== */

void scColumn::FlowsetInvalidateSelection( void )
{
    scColumn*       col = (scColumn*)FirstInChain();
    scSelection*    sel = col->GetSelection();

    if ( sel )
        sel->Invalidate();
}

/* ================================================================== */

void scColumn::RecomposeFlowset( long ticks, scRedispList* redispList )
{
    scColumn* col = (scColumn*)FirstInChain();

    SetRecomposition( true );

    for ( ; col; col = col->GetNext() ) {
        if ( col->Marked( scINVALID ) && col->DamOpen() )
            col->LimitDamage( redispList, ticks );
        else if ( col->Marked( scREALIGN ) ) {
            scRedisplayStoredLine rdl( GetLinecount( ) );
            rdl.SaveLineList( this );

            col->VertJustify();
        // pointer to first line
    pbuf = BufSet_long( pbuf, 0, kIntelOrder );


        // flow direction
    pbuf = BufSet_short( pbuf, (ushort)fFlowDir.GetLineDir(), kIntelOrder );
    pbuf = BufSet_short( pbuf, (ushort)fFlowDir.GetCharDir(), kIntelOrder );

        // width & depth
    pbuf = BufSet_long( pbuf, fSize.Width(), kIntelOrder );
    pbuf = BufSet_long( pbuf, fSize.Depth(), kIntelOrder );

        // application name
    pbuf = BufSet_long( pbuf,
                        APPPointerToDiskID( ctxPtr, fAppName, diskidColumn ),
                        kIntelOrder );

        // count
    pbuf = BufSet_long( pbuf, GetCount(), kIntelOrder  );

    scAssert ((size_t)(pbuf-abuf) == FILE_SIZE_COLUMN );

    WriteBytes( abuf, ctxPtr, writeFunc, FILE_SIZE_COLUMN );

    WriteLong( (ulong)fShapePieces, ctxPtr, writeFunc, kIntelOrder );

    switch ( GetShapeType() ) {

        default:
            break;

        case eVertShape:
            POLYtoFile( ctxPtr, writeFunc, fVertH, fShapePieces );
            break;

        case eRgnShape:
            RGNtoFile( ctxPtr, writeFunc, fRgnH, fShapePieces );
            break;
    }

    if ( !GetPrev() )
        fStream->STRToFile( ctxPtr, writeFunc );
}

/* ================================================================== */

void scColumn::RestorePointers( scSet* enumTable )
{
    if ( !Marked( scPTRRESTORED ) ) {
        scTBObj::RestorePointers( enumTable );

        AddToCTXList();

        fStream = (scStream*)enumTable->Get( (long)fStream );
        if ( !GetPrev() )
            fStream->STRRestorePointers( enumTable );
    }
}

/* ================================================================== */

void scColumn::SetRecomposition( Bool tf )
{
    scColumn* col = (scColumn*)FirstInChain();

    if ( tf )
        col->Mark( scLAYcomposeACTIVE );
    else
        col->Unmark( scLAYcomposeACTIVE );
}

/* ================================================================== */
```

```
    pbuf = BufGet_long( pbuf, uval, kIntelOrder );
    fStream = (scStream*)uval;

        // pointer to first line
    pbuf = BufGet_long( pbuf, uval, kIntelOrder );
    scAssert( uval == 0 );


        // flow direction
    ushort  uflow;
    pbuf = BufGet_short( pbuf, uflow, kIntelOrder );
    fFlowDir.SetLineDir( (eTextDirections)uflow );

    pbuf = BufGet_short( pbuf, uflow, kIntelOrder );
    fFlowDir.SetCharDir( (eTextDirections)uflow );

        // width & depth
    pbuf = BufGet_long( pbuf, uval, kIntelOrder );
    fSize.SetWidth( uval );

    pbuf = BufGet_long( pbuf, uval, kIntelOrder );
    fSize.SetDepth( uval );

        // application name
    pbuf = BufGet_long( pbuf, uval, kIntelOrder );
    fAppName = (APPColumn)APPDiskIDToPointer( ctxPtr, (long)uval, diskidColumn );

        // count
    pbuf = BufGet_long( pbuf, uval, kIntelOrder  );
    fColumnCount = uval;

    scAssert ((size_t)(pbuf-abuf) == FILE_SIZE_COLUMN );

        // shape type
    long val;
    ReadLong( val, ctxPtr, readFunc, kIntelOrder );

    if ( val ) {
        HRgnHandle rgnH = RGNfromFile( ctxPtr, readFunc, fShapePieces );

        SetShapeType( eRgnShape );

        fRgnH = rgnH;

        scAutoUnlock h( fRgnH );
        HRgn*   rgn = (HRgn *)*h;

        fShapePieces = (ushort)rgn->fNumSlivers;
    }
    else
        fShapePieces = 0;

    if ( !GetPrev() )
        scStream::STRFromFile( enumTable, ctxPtr, readFunc );
}

/* ================================================================== */
/* ACTUAL WRITE, this performs the write out of the column data structure,
 * paragraphs are written out with the first column in a set of linked columns
 * other than the column itself the only thing we will be writting out will
 * be the outline vertices
 */

void scColumn::Write( APPCtxPtr ctxPtr,
                      IOFuncPtr writeFunc )
{
    scTBObj::Write( ctxPtr, writeFunc );

    uchar   abuf[FILE_SIZE_COLUMN];
    uchar*  pbuf    = abuf;

        // pointer to stream
    pbuf = BufSet_long( pbuf, fStream ? fStream->GetEnumCount() : 0, kIntelOrder );
```

```
/***************************************************************************

    File:     SCCOLUMN.C

    $Header: /Projects/Toolbox/ct/Sccolumn.cpp 4     5/30/97 8:45a Wmanis $

    Contains:   The 'methods' for the column objects.

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

***************************************************************************/

#include "sccolumn.h"
#include "scpubobj.h"
#include "scapptex.h"
#include "sccallbk.h"
#include "scstcach.h"
#include "scglobda.h"
#include "scmem.h"
#include "scparagr.h"
#include "scpolygo.h"
#include "scregion.h"
#include "scselect.h"
#include "scstream.h"
#include "scset.h"
#include "sctextli.h"
#include "screfdat.h"
#include "scfileio.h"

#include <float.h>

/* ================================================================== */
/* ================================================================== */

scColumn* scColumn::fTheContextList = 0;

/* =================================================================== */

scColumn::~scColumn()
{
    delete fSelection, fSelection = 0;
}

/* =================================================================== */

#define FILE_SIZE_COLUMN    28

void scColumn::Read( scSet*     enumTable,
                     APPCtxPtr  ctxPtr,
                     IOFuncPtr  readFunc )
{
    uchar           abuf[FILE_SIZE_COLUMN];
    const uchar*    pbuf    = abuf;

    scTBObj::Read( enumTable, ctxPtr, readFunc );
    Mark( scINVALID );

        // read in the rest of the columns data
    ReadBytes( abuf, ctxPtr, readFunc, FILE_SIZE_COLUMN );

        // pointer to stream
    ulong uval;
```

```
#endif /* _H_SCCOLUMN */
```

```
                        }
    void            SetSelection( scSelection* sel )
                        {
                            fSelection = sel;
                        }

                        // actually allocate the real estate for lines
    virtual Bool    GetStrip2( scLINERefData&, int, scCOLRefData& );

private:

    static scColumn*    fTheContextList;

    void            CreateSelection( void );

    scColumn*       fNextContext;

    APPColumn       fAppName;       // application name

    long            fColumnCount;
#if 0
    MicroPoint      fWidth;         // width of column
    MicroPoint      fDepth;         // depth of column
#else
    scSize          fSize;
#endif

    scFlowDir       fFlowDir;       // the basic flow direction of a container

    scStream*       fStream;        // hook into stream
    scSelection*    fSelection;

    scXRect         fInkExtents;    // actual extents w/ italics, idents, etc.

    scTextline*     fFirstline;     // firstline of the column

    ushort          fShapePieces;   // num of components of shape
    union {
        scVertHandle    fVertH;
        HRgnHandle      fRgnH;
    };
};

/*======================================================================*/

#define FIRST_LINE_POSITION         (LONG_MIN + 1)
#define HorzFlexMeasure             (LONG_MAX - one_pica)

/* these seems arbitrary,
 * but we need to get it
 * away from LONG_MAX
 */

/* ================================================================== */
/*======================================================================*/


/* OPTIMIZATIONS */
#define COLShapePieces( c ) ( (c)->fShapePieces )



/* PROTOTYPES */
/*********************************************************************/
/*********************************************************************/

short       COLLineNum( scSelection* );


/*======================================================================*/
/*======================================================================*/
```

```
    void            AddToCTXList( )
                            {
                                fNextContext    = fTheContextList;
                                fTheContextList = this;
                            }
    void            DeleteFromCTXList(  );
    void            VerifyCTXList( void ) const;

    static void     ChangedTS( TypeSpec, eSpecTask, scRedispList* );
    static void     Update( scRedispList* );



    void            LineInfo( scLineInfoList*,
                              long&,
                              Bool& ) const;


    void            VertJustify( void );
    void            SetDepthNVJ( MicroPoint, scRedispList* );
    void            SetVJ( eVertJust );


                // COLUMN SELECTION
    void            ClickEvaluate( const scMuPoint&,
                                   REAL& );


    void            StartShiftClick( scStreamLocation&,
                                     const scMuPoint&,
                                     HiliteFuncPtr,
                                     APPDrwCtx,
                                     scSelection*& );

    void            StartClick( const scMuPoint&,
                                HiliteFuncPtr,
                                APPDrwCtx,
                                scSelection*& );

    void            ContinueClick( const scMuPoint& ,
                                   HiliteFuncPtr,
                                   scSelection* );

    Bool            Select( const scMuPoint&    hitPt,
                            TextMarker*         textMarker,
                            REAL*               bestDist );

    void            InitialSelection( TypeSpec&, scSelection*& );

    void            SelectSpecial( const scMuPoint&,
                                   eSelectModifier,
                                   scSelection*& );

    void            LimitDamage( scRedispList*, long );

    void            PasteAPPText( stTextImportExport&, scRedispList* );

    void            ReadTextFile( TypeSpec,
                                  APPCtxPtr,
                                  IOFuncPtr,
                                  scRedispList* );


protected:

                        // do not confuse the following with flowset operations
    scSelection*    GetSelection( void )
                        {
                            return fSelection;
                        }

    scColumn*       GetPrev( void ) const
                        {
                            return (scColumn*)Prev();
                        }
    scColumn*       GetNext( void ) const           { return (scColumn*)Next(); }

    void            SetCount( long count )
                        {
                            fColumnCount = count;
                        }
    long            GetCount( void ) const
                        {
                            return fColumnCount;
                        }

    void            SetFlowdir( const scFlowDir& fd )
                        {
                            fFlowDir = fd;
                        }
    const scFlowDir& GetFlowdir( void ) const
                        {
                            return fFlowDir;
                        }

    void            SetContext( scColumn* ctx )
                        {
                            fNextContext = ctx;
                        }
    scColumn*       GetContext( void ) const
                        {
                            return fNextContext;
                        }

    void            SetVertFlex( Bool, scRedispList* );
    void            SetHorzFlex( Bool, scRedispList* );
    Bool            GetVertFlex( void ) const
                        {
                            return GetShapeType() & eVertFlex;
                        }
    Bool            GetHorzFlex( void ) const
                        {
                            return GetShapeType() & eHorzFlex;
                        }


    void            Delete( scRedispList* );
    void            Free( void );
    void            FreeShape( void );
    void            FreeScrap( void );
    void            UpdateLine( scImmediateRedisp&, APPDrwCtx );

    void            LineExtents( scImmediateRedisp& );
    void            FreeLines( Bool, scXRect& );

    void            InvertExtents( HiliteFuncPtr, APPDrwCtx );
#if SCDEBUG > 1
    virtual void    scAssertValid( Bool recurse = true ) const;
    void            DbgPrintInfo( int debugLevel = 0 ) const;
#else
    virtual void    scAssertValid( Bool = true ) const{}
#endif

    static scColumn*    FindFlowset( const scStream* );

                    // context list
    static scColumn*    GetBaseContextList( void )
                        {
                            return fTheContextList;
                        }
    static void         FiniCTXList( void );
    eVertJust        GetVertJust( void ) const
                         {
                             return (eVertJust)fLayBits.fLayAdjustment;
                         }
    void             SetVertJust( eVertJust vj )
                         {
                             fLayBits.fLayAdjustment = vj;
                         }

    eColShapeType    GetShapeType( void ) const
                         {
                             return (eColShapeType)fLayBits.fLayType;
                         }
    void             SetShapeType( eColShapeType st );


    ushort           GetShapePieces( void ) const
                         {
                             return fShapePieces;
                         }

    scVertHandle     GetVertList( void ) const
                         {
                             return fVertH;
                         }
    void             SetVertList( scVertHandle vl )
                         {
                             fVertH = vl;
                         }

    HRgnHandle       GetRgn( void ) const
                         {
                             return fRgnH;
                         }
    void             SetRgn( HRgnHandle rgn )
                         {
                             fRgnH = rgn;
                         }

    void             SetAPPName( APPColumn appcol )
                         {
                             fAppName = appcol;
                         }
    APPColumn        GetAPPName( void ) const
                         {
                             return fAppName;
                         }

    void             SetWidth( MicroPoint w )
                         {
                             fSize.SetWidth( w );
                         }
    MicroPoint       Width( void ) const
                         {
                             return fSize.Width();
                         }

    void             SetDepth( MicroPoint d )        { fSize.SetDepth( d ); }
    MicroPoint       Depth( void ) const             { return fSize.Depth(); }

    void             SetSize( const scSize& size )
                         {
                             fSize = size;
                         }
    const scSize&    GetSize( void ) const
                         {
                             return fSize;
                         }

    void             SetSize( MicroPoint w, MicroPoint d )
                         {
                             fSize.SetWidth( w ), fSize.SetDepth( d );
    void            Rebreak( scRedispList* );
    void            Rebreak2( scRedispList* );

    void            ExternalSize( long& );
    void            ZeroEnumeration( void );

    Bool            GetStrip( scLINERefData&, int, scCOLRefData& );

    void            DeleteExcessLines( scContUnit*, scTextline*, Bool, scCOLRefData& );

    ///////////////////////////////////////////////////////////////////////
    //////////////////// COLUMN SHAPE METHODS /////////////////////////////
    ///////////////////////////////////////////////////////////////////////

    void            ReplacePoly( scVertHandle, scRedispList* );
    void            PastePoly( scVertHandle, scRedispList* );
    void            CopyPoly( scVertHandle* );

    void            PasteRgn( const HRgnHandle, scRedispList* );
    void            CopyRgn( HRgnHandle& );

    void            ClearShape( scRedispList* );



    ///////////////////////////////////////////////////////////////////////
    ////////////////// COLUMN LINKAGE METHODS /////////////////////////////
    ///////////////////////////////////////////////////////////////////////

    void            Link( scColumn*, Bool, scRedispList* );
    void            Unlink( scRedispList* );

    void            Renumber( void );

    void            BreakChain( scColumn* );

                        // get the next or previous column that
                        // actually contains lines (i.e. composed text )
                        //
    scColumn*       PrevWithLines( void ) const;
    scColumn*       NextWithLines( void ) const;

    void            ComputeInkExtents( void );

    void            SetInkExtents( MicroPoint x1, MicroPoint y1, MicroPoint x2, MicroPoint y2 )
                        {
                            fInkExtents.Set( x1, y1, x2, y2 );
                        }

    const scXRect&  GetInkExtents( void ) const
                        {
                            return fInkExtents;
                        }
    void            UnionInkExtents( const scXRect& xrect )
                        {
                            fInkExtents.Union( xrect );
                        }


    Bool            MoreText( void ) const;
    Bool            HasText( void ) const;

    scStream*       GetStream( void ) const
                        {
                            return fStream;
                        }
    void            SetStream( scStream* stream )
                        {
                            fStream = stream;
                        }
    void            SetFlowsetStream( scStream* stream );
    void            FreeStream( void );
```

```
                         APPDrwCtx,
                         const scMuPoint* translation = 0 );


    void            Hilite( const TextMarker&,
                            const TextMarker&,
                            HiliteFuncPtr,
                            const scSelection&  selection );


                        // FILE I/O

                        // complete the read
    virtual void    Read( scSet*, APPCtxPtr, IOFuncPtr );

                        // complete the write
    virtual void    Write( APPCtxPtr, IOFuncPtr );

                        // restore the pointers after completing a read
    virtual void    RestorePointers( scSet* );


    void            SetRecomposition( Bool tf );
    Bool            GetRecomposition( void ) const;


                        // get or set the first line of the column
                        //
    scTextline*     GetFirstline( void ) const
                        {
                            return fFirstline;
                        }
    void            SetFirstline( scTextline* txl )
                        {
                            fFirstline = txl;
                        }

    scTextline*     GetLastline( void ) const;


    void            TranslateLines( const scMuPoint& );
    void            RepositionLines( void );

    scContUnit*     MarkParas( void );
    scContUnit*     LastPara( void ) const;

                        // return the first paragraph in this container
                        // for reformatting purposes, we will assume that
                        // the previous container has been successfully
                        // reformatted
    scContUnit*     FirstPara( void ) const;



                        // return the number of lines for this column,
                        // if it is not formatted it will return -1
    ushort          GetLinecount( void ) const;

    virtual void    Resize( const scSize& size, scRedispList* = 0 );
    void            Resize( MicroPoint, MicroPoint, scRedispList* = 0 );

    scXRect&        RepaintExtent( scXRect& );
    void            QueryMargins( scXRect& ) const;
    void            QuerySize( scSize& ) const;
    void            QueryTextDepth( MicroPoint& ) const;
    MicroPoint      TextDepth( void ) const;
    void            GetTSList( scTypeSpecList& ) const;


            // should we reformat this column or wait till later
    Bool            DamOpen( void );
    /////////////////////////////////////////////////////////////////////
    ////////////////////// FLOW SET METHODS ////////////////////////////
    /////////////////////////////////////////////////////////////////////

                        // delete the stream from the flowset
                        // RETURNS the damaged area(s)
                        //
    void            FlowsetClearStream( scRedispList* );

                        // remove the stream from the flowset
                        // RETURNS the damaged area(s)
                        //
    void            FlowsetCutStream( scStream*, scRedispList*);

                        // paste the stream into the flowset
                        // RETURNS the damaged area(s)
                        //
    void            FlowsetPasteStream( scStream*, scRedispList* );


                        // get the selection object associated with
                        // the flowset, if there is none it will
                        // create one
                        //
    scSelection*    FlowsetGetSelection( void );

                        // set the selection object for the flowset
                        // none should exist, since if it does
                        // error recovery might be a bit tricky
                        //
    void            FlowsetSetSelection( scSelection* );

                        // this removes the selection from the flowset
                        // NOTE: it does not delete it
                        //
    void            FlowsetRemoveSelection( void );


                        // invalidate any selection associated with
                        // the flowset
    void            FlowsetInvalidateSelection( void );

                        // set the flow for the flowset
                        // all containers in a flowset must have the
                        // same flow at this time
    void            FlowsetSetFlowdir( const scFlowDir& );

    scColumn*       GetFlowset( void ) const
                        {
                            return (scColumn*)FirstInChain();
                        }

    void            RecomposeFlowset( long          ticks       = LONG_MAX,
                                      scRedispList* redisplist  = 0 );

    /////////////////////////////////////////////////////////////////////
    ////////////////////// COLUMN METHODS //////////////////////////////
    /////////////////////////////////////////////////////////////////////


    void            Enumerate( long& );



                        // draw the column updating the area
                        // intersected by the damage rect
                        //
    virtual void    Draw( const scXRect& damagedRect,
/***************************************************************************

     File:      SCCOLUMN.H


     $Header: /Projects/Toolbox/ct/SCCOLUMN.H 2     5/30/97 8:45a Wmanis $

     Contains:   text container definitions

     Written by: Manis

     Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
     All rights reserved.

     This notice is intended as a precaution against inadvertent publication
     and does not constitute an admission or acknowledgment that publication
     has occurred or constitute a waiver of confidentiality.

     Composition Toolbox software is the proprietary
     and confidential property of Stonehand Inc.

***************************************************************************/

#ifndef _H_SCCOLUMN
#define _H_SCCOLUMN

#include "sctbobj.h"


/* ================================================================== */
/* ================================================================== */
/* ================================================================== */

class scRedispList;
class scSelection;
class TextMarker;
class scImmediateRedisp;
class stTextImportExport;
class scLINERefData;
class scCOLRefData;
class scXRect;
class scRedispList;
class scTypeSpecList;
class scLineInfoList;
class scSpecLocList;
/* THE COLUMN OBJECT */

class scColumn : public scTBObj {
    scDECLARE_RTTI;

friend class scCOLRefData;

public:
                    scColumn( APPColumn,
                              MicroPoint,
                              MicroPoint,
                              scStream* p        = 0,
                              eCommonFlow   flow     = eRomanFlow );

                    scColumn() :
                        fShapePieces( 0 ),
                        fRgnH( 0 ),
                        fNextContext( 0 ),
                        fAppName( 0 ),
                        fColumnCount( 0 ),
                        fSize( 0, 0 ),
                        fFlowDir( eRomanFlow ),
                        fStream( 0 ),
                        fSelection( 0 ),
                        fFirstline( 0 ){}

                    ~scColumn();
    SCDebugTrace( 0, scString( "SCSPECLOCLIST\n" ) );
}
#endif

/* ==================================================================== */

TypeSpec scSpecLocList::GetLastValidSpec( void ) const
{
    for ( int i = NumItems() - 1; i >= 0; i-- ) {
        if ( (*this)[i].spec().ptr() )
            return (*this)[i].spec();
    }
    return 0;
}

/* ==================================================================== */

TypeSpec scSpecLocList::GetFirstValidSpec( void ) const
{
    for ( int i = 0; i < NumItems(); i++ ) {
        if ( (*this)[i].spec().ptr() )
            return (*this)[i].spec();
    }
    return 0;
}

/* ==================================================================== */

TypeSpec scSpecLocList::GetNthValidSpec( int nth ) const
{
    for ( int i = 0; i < NumItems(); i++ ) {
        if ( (*this)[i].spec().ptr() && --nth == 0 )
            return (*this)[i].spec();
    }
    return 0;
}

/* ==================================================================== */
```

```
    fPosOnLine( sl.fPosOnLine ),
    fSelMaxX( sl.fSelMaxX ),
    fFont( sl.fFont ),
    fPointSize( sl.fPointSize ),
    fBaseline( sl.fBaseline ),
    fMeasure( sl.fMeasure ),
    fLetterSpace( sl.fLetterSpace ),
    fWordSpace( sl.fWordSpace )
{
}

/* ====================================================================== */

scStreamLocation::scStreamLocation() :
    fStream( 0 ),
    fAPPColumn( 0 ),
    fParaNum( 0 ),
    fParaOffset( 0 ),
    fEndOfLine( 0 ),
    fTheCh( 0 ),
    fFlags( 0 ),
    fUnitType( eNoUnit ),
    fTheChWidth( 0 ),
    fChSpec( 0 ),
    fParaSpec( 0 ),
    fPosOnLine( 0 ),
    fSelMaxX( 0 ),
    fFont( 0 ),
    fPointSize( 0 ),
    fBaseline( 0 ),
    fMeasure( 0 ),
    fLetterSpace( 0 ),
    fWordSpace( 0 )
{
}

/* ====================================================================== */

void scStreamLocation::Init()
{
    fStream         = 0;
    fAPPColumn      = 0;
    fParaNum        = 0;
    fParaOffset     = 0;
    fEndOfLine      = 0;
    fTheCh          = 0;
    fFlags          = 0;
    fUnitType       = eNoUnit;
    fTheChWidth     = 0;
    fChSpec.clear();
    fParaSpec.clear();
    fPosOnLine      = 0;
    fSelMaxX        = 0;
    fFont           = 0;
    fPointSize      = 0;
    fBaseline       = 0;
    fMeasure        = 0;
    fLetterSpace    = 0;
    fWordSpace      = 0;
}

/* ====================================================================== */

#if SCDEBUG > 1
void scSpecLocList::DbgPrint( void ) const
{
    SCDebugTrace( 0, scString( "\nSCSPECLOCLIST\n" ) );
    for ( int i = 0; i < NumItems(); i++ ) {
        SCDebugTrace( 0, scString( "\tscCharSpecLoc ( %d %d ) 0x%08x\n" ),
                      (*this)[i].offset().fParaOffset,
                      (*this)[i].offset().fCharOffset,
                      (*this)[i].spec() );
    }
    fKeyCode         = rec.fKeyCode;
    field_           = rec.field_;
    fReplacedChar    = rec.fReplacedChar;
    replacedfield_   = rec.replacedfield_;
    fEscapement      = rec.fEscapement;
    fSpec            = rec.fSpec;
    fNoOp            = rec.fNoOp;
    fRestoreSelect   = rec.fRestoreSelect;
    fMark            = rec.fMark;

    return *this;
}

/* ================================================================== */

scKeyRecord::~scKeyRecord()
{
}

/* ================================================================== */

void scKeyRecord::Invert()
{
    UCS2 tmpChar = fReplacedChar;
    fReplacedChar = fKeyCode;
    fKeyCode = tmpChar;

    uint8 tmpfield = replacedfield_;
    replacedfield_ = field_;
    field_ = tmpfield;
}

/* ================================================================== */

scStreamLocation& scStreamLocation::operator=( const scStreamLocation& sl )
{
    fStream          = sl.fStream;
    fAPPColumn       = sl.fAPPColumn;
    fParaNum         = sl.fParaNum;
    fParaOffset      = sl.fParaOffset;
    fEndOfLine       = sl.fEndOfLine;
    fTheCh           = sl.fTheCh;
    fFlags           = sl.fFlags;
    fUnitType        = sl.fUnitType;
    fTheChWidth      = sl.fTheChWidth;
    fChSpec          = sl.fChSpec;
    fParaSpec        = sl.fParaSpec;
    fPosOnLine       = sl.fPosOnLine;
    fSelMaxX         = sl.fSelMaxX;
    fFont            = sl.fFont;
    fPointSize       = sl.fPointSize;
    fBaseline        = sl.fBaseline;
    fMeasure         = sl.fMeasure;
    fLetterSpace     = sl.fLetterSpace;
    fWordSpace       = sl.fWordSpace;

    return *this;
}

/* ================================================================== */

scStreamLocation::scStreamLocation( const scStreamLocation& sl ) :
    fStream( sl.fStream ),
    fAPPColumn( sl.fAPPColumn ),
    fParaNum( sl.fParaNum ),
    fParaOffset( sl.fParaOffset ),
    fEndOfLine( sl.fEndOfLine ),
    fTheCh( sl.fTheCh ),
    fFlags( sl.fFlags ),
    fUnitType( sl.fUnitType ),
    fTheChWidth( sl.fTheChWidth ),
    fChSpec( sl.fChSpec ),
    fParaSpec( sl.fParaSpec ),
/***************************************************************************

    File:       SCCSPECL.C


    $Header: /Projects/Toolbox/ct/SCCSPECL.CPP 2     5/30/97 8:45a Wmanis $

    Contains:   Maintains typespec list.

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.


***************************************************************************/

#include "scpubobj.h"

/* ================================================================== */

void scTypeSpecList::Insert( TypeSpec& ts )
{
    for ( int i = 0; i < NumItems(); i++ ) {
        if ( ts.ptr() == (*this)[i].ptr() )
            return;
    }
    Append( ts );
}

/* ================================================================== */

scKeyRecord::scKeyRecord() :
    type_( insert ),
    fKeyCode( 0 ),
    fReplacedChar( 0 ),
    field_( 0 ),
    replacedfield_( 0 ),
    fEscapement( 0 ),
    fSpec( 0 ),
    fNoOp( 0 ),
    fRestoreSelect( 0 )
{
}

/* ================================================================== */

scKeyRecord::scKeyRecord( const scKeyRecord& rec )
{
    type_           = rec.type_;
    fKeyCode        = rec.fKeyCode;
    field_          = rec.field_;
    fReplacedChar   = rec.fReplacedChar;
    replacedfield_  = rec.replacedfield_;
    fEscapement     = rec.fEscapement;
    fSpec           = rec.fSpec;
    fNoOp           = rec.fNoOp;
    fRestoreSelect  = rec.fRestoreSelect;
    fMark           = rec.fMark;
}

/* ================================================================== */

scKeyRecord& scKeyRecord::operator=( const scKeyRecord& rec )
{
    type_           = rec.type_;
```

```
/* ==================================================================== */
/* determines whether or not to store just the character and its flags or
 * the character, its flags and its escapement
 */

Bool CTStoreAll( UCS2   ch )
{
    switch ( ch ) {
        default:
            return false;
        case scFixAbsSpace:
        case scFixRelSpace:
            return true;
    }
}

/* ==================================================================== */
```

```
    sc_ALPHA|sc_LOCASE,             // [00FF] LATIN_SMALL_LETTER_Y_WITH_DIAERESIS

    0
};


/* ==================================================================== */

static UCS2 CTChangeCase( UCS2 );

/* ==================================================================== */
/* return the lower case of a character */

UCS2 CTToLower( UCS2 ch )
{
    register ushort test;

    if ( ch < 256 ) {
        test = sc_CharType[ch+1];
        if ( test & sc_UPCASE ) {
            if ( ch != 0xDF )
                return (UCS2)(ch + 0x20);
        }
    }
    else
        ; /* case may not be significant */

    return ch;
}

/* ==================================================================== */

UCS2 CTToUpper( register UCS2   ch )
{
    register ushort test;

    if ( ch < 256 ) {
        test = sc_CharType[ch+1];
        if ( test & sc_LOCASE ) {
            if ( ch != 0xFF )
                return (UCS2)(ch - 0x20);
        }
    }
    else
        ; /* case may not be significant */

    return ch;
}

/* ==================================================================== */

UCS2 CTToggleCase( register UCS2    ch )
{
    register ushort test;

    if ( ch < 256 ) {
        test = sc_CharType[ch+1];
        if ( test & sc_LOCASE ) {
            if ( ch != 0xFF )
                return (UCS2)(ch - 0x20);
        }
        else if ( test & sc_UPCASE ) {
            if ( ch != 0xDF )
                return (UCS2)(ch + 0x20);
        }
    }
    else
        ; /* case may not be significant */

    return ch;
}
```

```
    sc_SYMBOL,                      // [00BD] VULGAR_FRACTION_ONE_HALF
    sc_SYMBOL,                      // [00BE] VULGAR_FRACTION_THREE_QUARTERS
    sc_SYMBOL,                      // [00BF] INVERTED_QUESTION_MARK

    sc_ALPHA|sc_UPCASE,             // [00C0] LATIN_CAPITAL_LETTER_A_WITH_GRAVE
    sc_ALPHA|sc_UPCASE,             // [00C1] LATIN_CAPITAL_LETTER_A_WITH_ACUTE
    sc_ALPHA|sc_UPCASE,             // [00C2] LATIN_CAPITAL_LETTER_A_WITH_CIRCUMFLEX
    sc_ALPHA|sc_UPCASE,             // [00C3] LATIN_CAPITAL_LETTER_A_WITH_TILDE
    sc_ALPHA|sc_UPCASE,             // [00C4] LATIN_CAPITAL_LETTER_A_WITH_DIAERESIS
    sc_ALPHA|sc_UPCASE,             // [00C5] LATIN_CAPITAL_LETTER_A_WITH_RING_ABOVE
    sc_ALPHA|sc_UPCASE,             // [00C6] LATIN_CAPITAL_LIGATURE_AE
    sc_ALPHA|sc_UPCASE,             // [00C7] LATIN_CAPITAL_LETTER_C_WITH_CEDILLA
    sc_ALPHA|sc_UPCASE,             // [00C8] LATIN_CAPITAL_LETTER_E_WITH_GRAVE
    sc_ALPHA|sc_UPCASE,             // [00C9] LATIN_CAPITAL_LETTER_E_WITH_ACUTE
    sc_ALPHA|sc_UPCASE,             // [00CA] LATIN_CAPITAL_LETTER_E_WITH_CIRCUMFLEX
    sc_ALPHA|sc_UPCASE,             // [00CB] LATIN_CAPITAL_LETTER_E_WITH_DIAERESIS
    sc_ALPHA|sc_UPCASE,             // [00CC] LATIN_CAPITAL_LETTER_I_WITH_GRAVE
    sc_ALPHA|sc_UPCASE,             // [00CD] LATIN_CAPITAL_LETTER_I_WITH_ACUTE
    sc_ALPHA|sc_UPCASE,             // [00CE] LATIN_CAPITAL_LETTER_I_WITH_CIRCUMFLEX
    sc_ALPHA|sc_UPCASE,             // [00CF] LATIN_CAPITAL_LETTER_I_WITH_DIAERESIS
    sc_ALPHA|sc_UPCASE,             // [00D0] LATIN_CAPITAL_LETTER_ETH_(Icelandic)
    sc_ALPHA|sc_UPCASE,             // [00D1] LATIN_CAPITAL_LETTER_N_WITH_TILDE
    sc_ALPHA|sc_UPCASE,             // [00D2] LATIN_CAPITAL_LETTER_O_WITH_GRAVE
    sc_ALPHA|sc_UPCASE,             // [00D3] LATIN_CAPITAL_LETTER_O_WITH_ACUTE
    sc_ALPHA|sc_UPCASE,             // [00D4] LATIN_CAPITAL_LETTER_O_WITH_CIRCUMFLEX
    sc_ALPHA|sc_UPCASE,             // [00D5] LATIN_CAPITAL_LETTER_O_WITH_TILDE
    sc_ALPHA|sc_UPCASE,             // [00D6] LATIN_CAPITAL_LETTER_O_WITH_DIAERESIS

    sc_SYMBOL,                      // [00D7] MULTIPLICATION_SIGN

    sc_ALPHA|sc_UPCASE,             // [00D8] LATIN_CAPITAL_LETTER_O_WITH_STROKE
    sc_ALPHA|sc_UPCASE,             // [00D9] LATIN_CAPITAL_LETTER_U_WITH_GRAVE
    sc_ALPHA|sc_UPCASE,             // [00DA] LATIN_CAPITAL_LETTER_U_WITH_ACUTE
    sc_ALPHA|sc_UPCASE,             // [00DB] LATIN_CAPITAL_LETTER_U_WITH_CIRCUMFLEX
    sc_ALPHA|sc_UPCASE,             // [00DC] LATIN_CAPITAL_LETTER_U_WITH_DIAERESIS
    sc_ALPHA|sc_UPCASE,             // [00DD] LATIN_CAPITAL_LETTER_Y_WITH_ACUTE
    sc_ALPHA|sc_UPCASE,             // [00DE] LATIN_CAPITAL_LETTER_THORN_(Icelandic)

    sc_ALPHA|sc_LOCASE,             // [00DF] LATIN_SMALL_LETTER_SHARP_S_(German)

    sc_ALPHA|sc_LOCASE,             // [00E0] LATIN_SMALL_LETTER_A_WITH_GRAVE
    sc_ALPHA|sc_LOCASE,             // [00E1] LATIN_SMALL_LETTER_A_WITH_ACUTE
    sc_ALPHA|sc_LOCASE,             // [00E2] LATIN_SMALL_LETTER_A_WITH_CIRCUMFLEX
    sc_ALPHA|sc_LOCASE,             // [00E3] LATIN_SMALL_LETTER_A_WITH_TILDE
    sc_ALPHA|sc_LOCASE,             // [00E4] LATIN_SMALL_LETTER_A_WITH_DIAERESIS
    sc_ALPHA|sc_LOCASE,             // [00E5] LATIN_SMALL_LETTER_A_WITH_RING_ABOVE
    sc_ALPHA|sc_LOCASE,             // [00E6] LATIN_SMALL_LIGATURE_AE
    sc_ALPHA|sc_LOCASE,             // [00E7] LATIN_SMALL_LETTER_C_WITH_CEDILLA
    sc_ALPHA|sc_LOCASE,             // [00E8] LATIN_SMALL_LETTER_E_WITH_GRAVE
    sc_ALPHA|sc_LOCASE,             // [00E9] LATIN_SMALL_LETTER_E_WITH_ACUTE
    sc_ALPHA|sc_LOCASE,             // [00EA] LATIN_SMALL_LETTER_E_WITH_CIRCUMFLEX
    sc_ALPHA|sc_LOCASE,             // [00EB] LATIN_SMALL_LETTER_E_WITH_DIAERESIS
    sc_ALPHA|sc_LOCASE,             // [00EC] LATIN_SMALL_LETTER_I_WITH_GRAVE
    sc_ALPHA|sc_LOCASE,             // [00ED] LATIN_SMALL_LETTER_I_WITH_ACUTE
    sc_ALPHA|sc_LOCASE,             // [00EE] LATIN_SMALL_LETTER_I_WITH_CIRCUMFLEX
    sc_ALPHA|sc_LOCASE,             // [00EF] LATIN_SMALL_LETTER_I_WITH_DIAERESIS
    sc_ALPHA|sc_LOCASE,             // [00F0] LATIN_SMALL_LETTER_ETH_(Icelandic)
    sc_ALPHA|sc_LOCASE,             // [00F1] LATIN_SMALL_LETTER_N_WITH_TILDE
    sc_ALPHA|sc_LOCASE,             // [00F2] LATIN_SMALL_LETTER_O_WITH_GRAVE
    sc_ALPHA|sc_LOCASE,             // [00F3] LATIN_SMALL_LETTER_O_WITH_ACUTE
    sc_ALPHA|sc_LOCASE,             // [00F4] LATIN_SMALL_LETTER_O_WITH_CIRCUMFLEX
    sc_ALPHA|sc_LOCASE,             // [00F5] LATIN_SMALL_LETTER_O_WITH_TILDE
    sc_ALPHA|sc_LOCASE,             // [00F6] LATIN_SMALL_LETTER_O_WITH_DIAERESIS

    sc_SYMBOL,                      // [00F7] DIVISION_SIGN

    sc_ALPHA|sc_LOCASE,             // [00F8] LATIN_SMALL_LETTER_O_WITH_STROKE
    sc_ALPHA|sc_LOCASE,             // [00F9] LATIN_SMALL_LETTER_U_WITH_GRAVE
    sc_ALPHA|sc_LOCASE,             // [00FA] LATIN_SMALL_LETTER_U_WITH_ACUTE
    sc_ALPHA|sc_LOCASE,             // [00FB] LATIN_SMALL_LETTER_U_WITH_CIRCUMFLEX
    sc_ALPHA|sc_LOCASE,             // [00FC] LATIN_SMALL_LETTER_U_WITH_DIAERESIS
    sc_ALPHA|sc_LOCASE,             // [00FD] LATIN_SMALL_LETTER_Y_WITH_ACUTE
    sc_ALPHA|sc_LOCASE,             // [00FE] LATIN_SMALL_LETTER_THORN_(Icelandic)
    sc_ASCII|sc_ALPHA|sc_LOCASE,        // [0076] LATIN_SMALL_LETTER_V
    sc_ASCII|sc_ALPHA|sc_LOCASE,        // [0077] LATIN_SMALL_LETTER_W
    sc_ASCII|sc_ALPHA|sc_LOCASE,        // [0078] LATIN_SMALL_LETTER_X
    sc_ASCII|sc_ALPHA|sc_LOCASE,        // [0079] LATIN_SMALL_LETTER_Y
    sc_ASCII|sc_ALPHA|sc_LOCASE,        // [007A] LATIN_SMALL_LETTER_Z
    sc_ASCII|sc_SYMBOL,                 // [007B] LEFT_CURLY_BRACKET
    sc_ASCII|sc_SYMBOL,                 // [007C] VERTICAL_LINE
    sc_ASCII|sc_SYMBOL,                 // [007D] RIGHT_CURLY_BRACKET
    sc_ASCII|sc_SYMBOL,                 // [007E] TILDE

    0,                              // 0x7f 127
    0,                              // 0x80 128
    0,                              // 0x81 129
    0,                              // 0x82 130
    0,                              // 0x83 131
    0,                              // 0x84 132
    0,                              // 0x85 133
    0,                              // 0x86 134
    0,                              // 0x87 135
    0,                              // 0x88 136
    0,                              // 0x89 137
    0,                              // 0x8a 138
    0,                              // 0x8b 139
    0,                              // 0x8c 140
    0,                              // 0x8d 141
    0,                              // 0x8e 142
    0,                              // 0x8f 143
    0,                              // 0x90 144
    0,                              // 0x91 145
    0,                              // 0x92 146
    0,                              // 0x93 147
    0,                              // 0x94 148
    0,                              // 0x95 149
    0,                              // 0x96 150
    0,                              // 0x97 151
    0,                              // 0x98 152
    0,                              // 0x99 153
    0,                              // 0x9a 154
    0,                              // 0x9b 155
    0,                              // 0x9c 156
    0,                              // 0x9d 157
    0,                              // 0x9e 158
    0,                              // 0x9f 159

    sc_SPACE,                       // [00A0] NO-BREAK_SPACE
    sc_SYMBOL,                      // [00A1] INVERTED_EXCLAMATION_MARK
    sc_SYMBOL,                      // [00A2] CENT_SIGN
    sc_SYMBOL,                      // [00A3] POUND_SIGN
    sc_SYMBOL,                      // [00A4] CURRENCY_SIGN
    sc_SYMBOL,                      // [00A5] YEN_SIGN
    sc_SYMBOL,                      // [00A6] BROKEN_BAR
    sc_SYMBOL,                      // [00A7] SECTION_SIGN
    sc_SYMBOL,                      // [00A8] DIAERESIS
    sc_SYMBOL,                      // [00A9] COPYRIGHT_SIGN
    sc_SYMBOL,                      // [00AA] FEMININE_ORDINAL_INDICATOR
    sc_SYMBOL,                      // [00AB] LEFT-POINTING_DOUBLE_ANGLE_QUOTATION_MARK
    sc_SYMBOL,                      // [00AC] NOT_SIGN
    sc_SYMBOL,                      // [00AD] SOFT_HYPHEN
    sc_SYMBOL,                      // [00AE] REGISTERED_SIGN
    sc_SYMBOL,                      // [00AF] MACRON
    sc_SYMBOL,                      // [00B0] DEGREE_SIGN
    sc_SYMBOL,                      // [00B1] PLUS-MINUS_SIGN
    sc_SYMBOL,                      // [00B2] SUPERSCRIPT_TWO
    sc_SYMBOL,                      // [00B3] SUPERSCRIPT_THREE
    sc_SYMBOL,                      // [00B4] ACUTE_ACCENT
    sc_SYMBOL,                      // [00B5] MICRO_SIGN
    sc_SYMBOL,                      // [00B6] PILCROW_SIGN
    sc_SYMBOL,                      // [00B7] MIDDLE_DOT
    sc_SYMBOL,                      // [00B8] CEDILLA
    sc_SYMBOL,                      // [00B9] SUPERSCRIPT_ONE
    sc_SYMBOL,                      // [00BA] MASCULINE_ORDINAL_INDICATOR
    sc_SYMBOL,                      // [00BB] RIGHT-POINTING_DOUBLE_ANGLE_QUOTATION_MARK
    sc_SYMBOL,                      // [00BC] VULGAR_FRACTION_ONE_QUARTER
    sc_ASCII|sc_PUNC,                   // [002D] HYPHEN-MINUS
    sc_ASCII|sc_PUNC,                   // [002E] FULL_STOP
    sc_ASCII|sc_SYMBOL,                 // [002F] SOLIDUS
    sc_ASCII|sc_DIGIT,                  // [0030] DIGIT_ZERO
    sc_ASCII|sc_DIGIT,                  // [0031] DIGIT_ONE
    sc_ASCII|sc_DIGIT,                  // [0032] DIGIT_TWO
    sc_ASCII|sc_DIGIT,                  // [0033] DIGIT_THREE
    sc_ASCII|sc_DIGIT,                  // [0034] DIGIT_FOUR
    sc_ASCII|sc_DIGIT,                  // [0035] DIGIT_FIVE
    sc_ASCII|sc_DIGIT,                  // [0036] DIGIT_SIX
    sc_ASCII|sc_DIGIT,                  // [0037] DIGIT_SEVEN
    sc_ASCII|sc_DIGIT,                  // [0038] DIGIT_EIGHT
    sc_ASCII|sc_DIGIT,                  // [0039] DIGIT_NINE
    sc_ASCII|sc_PUNC,                   // [003A] COLON
    sc_ASCII|sc_PUNC,                   // [003B] SEMICOLON
    sc_ASCII|sc_SYMBOL,                 // [003C] LESS-THAN_SIGN
    sc_ASCII|sc_SYMBOL,                 // [003D] EQUALS_SIGN
    sc_ASCII|sc_SYMBOL,                 // [003E] GREATER-THAN_SIGN
    sc_ASCII|sc_PUNC,                   // [003F] QUESTION_MARK
    sc_ASCII|sc_SYMBOL,                 // [0040] COMMERCIAL_AT
    sc_ASCII|sc_ALPHA|sc_UPCASE,        // [0041] LATIN_CAPITAL_LETTER_A
    sc_ASCII|sc_ALPHA|sc_UPCASE,        // [0042] LATIN_CAPITAL_LETTER_B
    sc_ASCII|sc_ALPHA|sc_UPCASE,        // [0043] LATIN_CAPITAL_LETTER_C
    sc_ASCII|sc_ALPHA|sc_UPCASE,        // [0044] LATIN_CAPITAL_LETTER_D
    sc_ASCII|sc_ALPHA|sc_UPCASE,        // [0045] LATIN_CAPITAL_LETTER_E
    sc_ASCII|sc_ALPHA|sc_UPCASE,        // [0046] LATIN_CAPITAL_LETTER_F
    sc_ASCII|sc_ALPHA|sc_UPCASE,        // [0047] LATIN_CAPITAL_LETTER_G
    sc_ASCII|sc_ALPHA|sc_UPCASE,        // [0048] LATIN_CAPITAL_LETTER_H
    sc_ASCII|sc_ALPHA|sc_UPCASE,        // [0049] LATIN_CAPITAL_LETTER_I
    sc_ASCII|sc_ALPHA|sc_UPCASE,        // [004A] LATIN_CAPITAL_LETTER_J
    sc_ASCII|sc_ALPHA|sc_UPCASE,        // [004B] LATIN_CAPITAL_LETTER_K
    sc_ASCII|sc_ALPHA|sc_UPCASE,        // [004C] LATIN_CAPITAL_LETTER_L
    sc_ASCII|sc_ALPHA|sc_UPCASE,        // [004D] LATIN_CAPITAL_LETTER_M
    sc_ASCII|sc_ALPHA|sc_UPCASE,        // [004E] LATIN_CAPITAL_LETTER_N
    sc_ASCII|sc_ALPHA|sc_UPCASE,        // [004F] LATIN_CAPITAL_LETTER_O
    sc_ASCII|sc_ALPHA|sc_UPCASE,        // [0050] LATIN_CAPITAL_LETTER_P
    sc_ASCII|sc_ALPHA|sc_UPCASE,        // [0051] LATIN_CAPITAL_LETTER_Q
    sc_ASCII|sc_ALPHA|sc_UPCASE,        // [0052] LATIN_CAPITAL_LETTER_R
    sc_ASCII|sc_ALPHA|sc_UPCASE,        // [0053] LATIN_CAPITAL_LETTER_S
    sc_ASCII|sc_ALPHA|sc_UPCASE,        // [0054] LATIN_CAPITAL_LETTER_T
    sc_ASCII|sc_ALPHA|sc_UPCASE,        // [0055] LATIN_CAPITAL_LETTER_U
    sc_ASCII|sc_ALPHA|sc_UPCASE,        // [0056] LATIN_CAPITAL_LETTER_V
    sc_ASCII|sc_ALPHA|sc_UPCASE,        // [0057] LATIN_CAPITAL_LETTER_W
    sc_ASCII|sc_ALPHA|sc_UPCASE,        // [0058] LATIN_CAPITAL_LETTER_X
    sc_ASCII|sc_ALPHA|sc_UPCASE,        // [0059] LATIN_CAPITAL_LETTER_Y
    sc_ASCII|sc_ALPHA|sc_UPCASE,        // [005A] LATIN_CAPITAL_LETTER_Z
    sc_ASCII|sc_SYMBOL,                 // [005B] LEFT_SQUARE_BRACKET
    sc_ASCII|sc_SYMBOL,                 // [005C] REVERSE_SOLIDUS
    sc_ASCII|sc_SYMBOL,                 // [005D] RIGHT_SQUARE_BRACKET
    sc_ASCII|sc_SYMBOL,                 // [005E] CIRCUMFLEX_ACCENT
    sc_ASCII|sc_SYMBOL,                 // [005F] LOW_LINE
    sc_ASCII|sc_ACCENT,                 // [0060] GRAVE_ACCENT
    sc_ASCII|sc_ALPHA|sc_LOCASE,        // [0061] LATIN_SMALL_LETTER_A
    sc_ASCII|sc_ALPHA|sc_LOCASE,        // [0062] LATIN_SMALL_LETTER_B
    sc_ASCII|sc_ALPHA|sc_LOCASE,        // [0063] LATIN_SMALL_LETTER_C
    sc_ASCII|sc_ALPHA|sc_LOCASE,        // [0064] LATIN_SMALL_LETTER_D
    sc_ASCII|sc_ALPHA|sc_LOCASE,        // [0065] LATIN_SMALL_LETTER_E
    sc_ASCII|sc_ALPHA|sc_LOCASE,        // [0066] LATIN_SMALL_LETTER_F
    sc_ASCII|sc_ALPHA|sc_LOCASE,        // [0067] LATIN_SMALL_LETTER_G
    sc_ASCII|sc_ALPHA|sc_LOCASE,        // [0068] LATIN_SMALL_LETTER_H
    sc_ASCII|sc_ALPHA|sc_LOCASE,        // [0069] LATIN_SMALL_LETTER_I
    sc_ASCII|sc_ALPHA|sc_LOCASE,        // [006A] LATIN_SMALL_LETTER_J
    sc_ASCII|sc_ALPHA|sc_LOCASE,        // [006B] LATIN_SMALL_LETTER_K
    sc_ASCII|sc_ALPHA|sc_LOCASE,        // [006C] LATIN_SMALL_LETTER_L
    sc_ASCII|sc_ALPHA|sc_LOCASE,        // [006D] LATIN_SMALL_LETTER_M
    sc_ASCII|sc_ALPHA|sc_LOCASE,        // [006E] LATIN_SMALL_LETTER_N
    sc_ASCII|sc_ALPHA|sc_LOCASE,        // [006F] LATIN_SMALL_LETTER_O
    sc_ASCII|sc_ALPHA|sc_LOCASE,        // [0070] LATIN_SMALL_LETTER_P
    sc_ASCII|sc_ALPHA|sc_LOCASE,        // [0071] LATIN_SMALL_LETTER_Q
    sc_ASCII|sc_ALPHA|sc_LOCASE,        // [0072] LATIN_SMALL_LETTER_R
    sc_ASCII|sc_ALPHA|sc_LOCASE,        // [0073] LATIN_SMALL_LETTER_S
    sc_ASCII|sc_ALPHA|sc_LOCASE,        // [0074] LATIN_SMALL_LETTER_T
    sc_ASCII|sc_ALPHA|sc_LOCASE,        // [0075] LATIN_SMALL_LETTER_U
```

```
/*****************************************************************************

     File:       SCCTYPE.C


     $Header: /Projects/Toolbox/ct/SCCTYPE.CPP 2     5/30/97 8:45a Wmanis $

     Contains:   Character types.

     Written by: Manis

     Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
     All rights reserved.

     This notice is intended as a precaution against inadvertent publication
     and does not constitute an admission or acknowledgment that publication
     has occurred or constitute a waiver of confidentiality.

     Composition Toolbox software is the proprietary
     and confidential property of Stonehand Inc.

*****************************************************************************/

#include "sccharex.h"
#include "scctype.h"

unsigned short sc_CharType[258] = {
    0,                                  /* to let us use -1 as an index */
    0,                                  /* 0x00 0 " " */
    0,                                  /* 0x01 1 " " */
    0,                                  /* 0x02 2 " " */
    0,                                  /* 0x03 3 " " */
    0,                                  /* 0x04 4 " " */
    0,                                  /* 0x05 5 " " */
    0,                                  /* 0x06 6 " " */
    sc_ASCII|sc_SPACE,                  /* 0x07 7 "indent space" */
    0,                                  /* 0x08 8 ""  */
    sc_ASCII|sc_SPACE,                  /* 0x09   9 "\t" (HT) tab key */
    sc_ASCII|sc_SPACE,                  /* 0x0a  10 "\n" (LF) line feed */
    sc_ASCII|sc_SPACE,                  /* 0x0b  11 "\l"  (VT) vertical tab */
    0,                                  /* 0x0c  12 " " */
    sc_ASCII|sc_SPACE,                  /* 0x0d  13 "\r" (CR) return key*/
    0,                                  /* 0x0e  14 " " */
    0,                                  /* 0x0f  15 " " */
    0,                                  /* 0x10  16 " " */
    0,                                  /* 0x11  17 " " */
    sc_ASCII,                           /* 0x12  18 "paraEnd" */
    sc_ASCII|sc_SPACE,                  /* 0x13  19 "quad center" */
    sc_ASCII|sc_SPACE,                  /* 0x14  20 "quad left" */
    sc_ASCII|sc_SPACE,                  /* 0x15  21 "quad right" */
    sc_ASCII|sc_SPACE,                  /* 0x16  22 "quad just" */
    sc_ASCII|sc_SPACE,                  /* 0x17  23 " " fix abs space */
    sc_ASCII|sc_SPACE,                  /* 0x18  24 " " fix rel space */
    sc_ASCII|sc_SPACE,                  /* 0x19  25 " " fill space */
    sc_ASCII,                           /* 0x1a  26 " " no break hyphen */
    sc_ASCII|sc_SPACE,                  /* 0x1b  27 " " discretionary hyphen */
    sc_ASCII|sc_SPACE,                  /* 0x1c  28 " " figure space */
    sc_ASCII|sc_SPACE,                  /* 0x1d  29 " " thin space */
    sc_ASCII|sc_SPACE,                  /* 0x1e  30 " " en space */
    sc_ASCII|sc_SPACE,                  /* 0x1f  31 " " em space */
    sc_ASCII|sc_SPACE,                  // [0020] SPACE
    sc_ASCII|sc_PUNC,                   // [0021] EXCLAMATION_MARK
    sc_ASCII|sc_PUNC,                   // [0022] QUOTATION_MARK
    sc_ASCII|sc_SYMBOL,                 // [0023] NUMBER_SIGN
    sc_ASCII|sc_SYMBOL,                 // [0024] DOLLAR_SIGN
    sc_ASCII|sc_SYMBOL,                 // [0025] PERCENT_SIGN
    sc_ASCII|sc_SYMBOL,                 // [0026] AMPERSAND
    sc_ASCII|sc_PUNC,                   // [0027] APOSTROPHE
    sc_ASCII|sc_SYMBOL,                 // [0028] LEFT_PARENTHESIS
    sc_ASCII|sc_SYMBOL,                 // [0029] RIGHT_PARENTHESIS
    sc_ASCII|sc_SYMBOL,                 // [002A] ASTERISK
    sc_ASCII|sc_SYMBOL,                 // [002B] PLUS_SIGN
    sc_ASCII|sc_PUNC,                   // [002C] COMMA
```

```
        // since the get strip logic uses regions and can only really
        // deal in one dimension we need to convert the coordinate
        // system of the used variables as we go in and out of the
        // get strip code - refer to the discussion of coordinate
        // systems in the Toolbox Concept doc
    if ( fPData.fComposedLine.IsVertical() ) {
        fPData.fComposedLine.fLogicalExtents.FourthToThird( 0 );
    }

    fPData.fComposedLine.fOrg.y = fPData.fComposedLine.fBaseline;

    doit = fCol->GetStrip2( fPData.fComposedLine, fPData.fBreakType, *this );

    fPData.fComposedLine.fBaseline  = fPData.fComposedLine.fOrg.y;

    if ( fPData.fComposedLine.IsVertical() ) {
        fPData.fComposedLine.fOrg.ThirdToFourth( fCol->Width() );
        fPData.fComposedLine.fLogicalExtents.ThirdToFourth( 0 );
    }

    return doit;
}

/* ==================================================================== */
```

09629370 .073400

```
).GetSpec() );
            else
                lineData.fOrg.y += lineData.fInitialLead.GetLead();

            lineData.fMeasure = colWidth - scCachedStyle::GetParaStyle().GetLeftBlockIndent();
            return lineData.fMeasure > 0;

        case eHorzFlex:
            lineData.fOrg.x         =  scCachedStyle::GetParaStyle().GetLeftBlockIndent();
            if ( lineData.fOrg.y == FIRST_LINE_POSITION )
                lineData.fOrg.y = CSfirstLinePosition( GetAPPName(), scCachedStyle::GetCurrentCache(
).GetSpec() );
            else
                lineData.fOrg.y += lineData.fInitialLead.GetLead();
            lineData.fMeasure       = HorzFlexMeasure;

            return lineData.fOrg.y <= colDepth - CSlastLinePosition( GetAPPName(), scCachedStyle::Ge
tCurrentCache().GetSpec() );

        case eFlexShape:
            lineData.fOrg.x         =  scCachedStyle::GetParaStyle().GetLeftBlockIndent();
            if ( lineData.fOrg.y == FIRST_LINE_POSITION )
                lineData.fOrg.y = CSfirstLinePosition( GetAPPName(), scCachedStyle::GetCurrentCache(
).GetSpec() );
            else
                lineData.fOrg.y += lineData.fInitialLead.GetLead();
            lineData.fMeasure       = HorzFlexMeasure;

            return true;
    }
    /*NOTREACHED*/
    return false;
}

/* ================================================================== */
// allocate geometry using args

Bool scColumn::GetStrip( scLINERefData& lineData,
                         int            breakType,
                         scCOLRefData&  colRefData )
{
    Bool        doit;

        // since the get strip logic uses regions and can only really
        // deal in one dimension we need to convert the coordinate
        // system of the used variables as we go in and out of the
        // get strip code - refer to the discussion of coordinate
        // systems in the Toolbox Concept doc
    if ( lineData.IsVertical() )
        lineData.fLogicalExtents.FourthToThird( 0 );

    lineData.fOrg.y     = lineData.fBaseline;

    doit = GetStrip2( lineData, breakType, colRefData );

    lineData.fBaseline  = lineData.fOrg.y;

    if ( lineData.IsVertical() ) {
        lineData.fOrg.ThirdToFourth( Width() );
        lineData.fLogicalExtents.ThirdToFourth( 0 );
    }

    return doit;
}


/* ================================================================== */
// allocate geeomtry using cached values

Bool scCOLRefData::AllocGeometry( void )
{
    Bool        doit;
```

```
.fInitialLead.GetLead() );
                    tryX                = colRefData.fPrevEnd.x;
                    tryY                = lineData.fOrg.y - firstLinePosition;
                    tryRect.y2          = firstLinePosition + CSlastLinePosition( GetAPPName(), scCa
chedStyle::GetCurrentCache().GetSpec() );
                }
                else
                    tryY = lineData.fOrg.y + lineData.fLogicalExtents.y1;

                if ( lineData.IsHorizontal() ) {
                    if ( colRefData.fPrevEnd.y == lineData.fOrg.y )
                        tryX = colRefData.fPrevEnd.x;
                    else
                        tryX = LONG_MIN;
                }
                else {
                    if ( ( colDepth - colRefData.fPrevEnd.x ) == lineData.fOrg.y )
                        tryX = colRefData.fPrevEnd.y;
                    else
                        tryX = LONG_MIN;
                }
            }

            colRefData.fRgn->SectRect( tryRect, tryY, tryX, lineData.fInitialLead.GetLead() );

            if ( lineData.fOrg.y == FIRST_LINE_POSITION || lineData.fOrg.y == colRefData.GetFirstlin
ePos() ) {
                    // this is here to fix the smi bug 1538 - given that we are using approximations

                    // alot in regions this may be an insufficient fix for other issues that smi
                    // may raise, but since we are using approximations i have no way of reliably
                    // pridicting these issues
                scXRect rgnXRect( colRefData.fRgn->fOrigBounds );
                scXRect tryXRect( tryRect );

                if ( !rgnXRect.Contains( tryXRect ) )
                    tryRect.x2 = tryRect.x1;
            }

            if ( tryRect.Width() == 0 )
                return false;
            else {
                if ( firstLine == true )
                    lineData.fOrg.y = tryRect.y1 + firstLinePosition;
                else
                    lineData.fOrg.y = tryRect.y1 - lineData.fLogicalExtents.y1;

                if ( lineData.fOrg.y > RGNMaxDepth( colRefData.fRgnH ) )
                    return false;

#if defined(LEFTBLOCKINDENT)
                if ( tryRect.x < 0 )
                    lineData.fOrg.x = tryRect.x +  gfmS.GetLeftBlockIndent();
                else
                    lineData.fOrg.x = MAX( tryRect.x,  gfmS.GetLeftBlockIndent() );
#else
                lineData.fOrg.x         = tryRect.x1 +  scCachedStyle::GetParaStyle().GetLeftBlockIn
dent();
#endif


                if ( lineData.fOrg.x != tryRect.x1 )
                    lineData.fMeasure = tryRect.Width() + ( tryRect.x1 - lineData.fOrg.x );
                else
                    lineData.fMeasure = tryRect.Width();

                return true;
            }
            break;    /*NOTREACHED*/

        case eVertFlex:
            lineData.fOrg.x         =  scCachedStyle::GetParaStyle().GetLeftBlockIndent();
            if ( lineData.fOrg.y == FIRST_LINE_POSITION )
                lineData.fOrg.y = CSfirstLinePosition( GetAPPName(), scCachedStyle::GetCurrentCache(
    Bool            firstLine   = false;
    int             shapeType;

        // the specs have been properly inited so the block
        // indent values should be correct
    scAssert( this ==  colRefData.GetActive() );

        // we are in an overflow condidtion
    if ( lineData.fOrg.y == LONG_MIN )
        return false;
    if ( breakType == eColumnBreak )
        return false;

    lineData.fColShapeType  = GetShapeType();
    shapeType               = GetShapeType();

        // We ran into a memory error in COLStartReformat; just use rectangle shape.
    if (fRgnH && colRefData.fRgnH == NULL )
        lineData.fColShapeType = eNoShape;

    if ( lineData.IsVertical() ) {
        colWidth    = Depth();
        colDepth    = Width();
        switch ( GetShapeType() ) {
            case eVertFlex:
                shapeType = eHorzFlex;
                break;
            case eHorzFlex:
                shapeType = eVertFlex;
                break;
        }
    }

    colRefData.fPrevEnd     = colRefData.fSavedPrevEnd;

    switch ( shapeType ) {
        default:
        case eNoShape:
            lineData.fOrg.x     =  scCachedStyle::GetParaStyle().GetLeftBlockIndent();
            lineData.fMeasure   = colWidth - scCachedStyle::GetParaStyle().GetLeftBlockIndent();
            if ( lineData.fOrg.y == FIRST_LINE_POSITION )
                lineData.fOrg.y = CSfirstLinePosition( GetAPPName(), scCachedStyle::GetCurrentCache(
).GetSpec() );
            else
                lineData.fOrg.y += lineData.fInitialLead.GetLead();

            return lineData.fOrg.y <= colDepth - CSlastLinePosition( GetAPPName(), scCachedStyle::Ge
tCurrentCache().GetSpec() );

        case eVertShape:
        case eRgnShape:
            tryRect.Set( 0, 0, MAX( scCachedStyle::GetParaStyle().GetMinMeasure(), colRefData.fRgn->
fVertInterval ), lineData.fLogicalExtents.Depth() );

            if ( lineData.fOrg.y == FIRST_LINE_POSITION ) {
                firstLine           = true;
                firstLinePosition   = CSfirstLinePosition( GetAPPName(), scCachedStyle::GetCurrentCa
che().GetSpec()  );
                lineData.fOrg.y     = colRefData.fRgn->FirstLinePos( firstLinePosition, lineData.fIn
itialLead.GetLead() );
                tryX                = colRefData.fPrevEnd.x;
                tryY                = lineData.fOrg.y - firstLinePosition;
                tryRect.y2          = firstLinePosition + CSlastLinePosition( GetAPPName(), scCached
Style::GetCurrentCache().GetSpec() );
                colRefData.SetFirstlinePos( lineData.fOrg.y );
                colRefData.SetFirstSpec( scCachedStyle::GetCurrentCache().GetSpec() );
            }
            else {
                if ( lineData.fOrg.y == colRefData.GetFirstlinePos() ) {
                    firstLine           = true;
                    firstLinePosition   = CSfirstLinePosition( GetAPPName(), colRefData.GetFirstSpec
() );
                    lineData.fOrg.y     = colRefData.fRgn->FirstLinePos( firstLinePosition, lineData
                    }
                }
                else {
                    if ( fCol->GetShapeType() & eHorzFlex ) {
                        scMuPoint trans( extents.Width() - fCol->Width(), 0 );
                        fCol->TranslateLines( trans );

                        fCol->SetWidth( extents.Width() );
                    }
                    if ( fCol->GetShapeType() & eVertFlex ) {
                        fCol->SetDepth( extents.y2 );
                        fCol->RepositionLines();
                    }
                }
                fCol->VertJustify();
                break;
        }

        fSavedLineState.LineListChanges( fCol, fLineDamage, fRedispList );

        if ( finished )
            fCol->Unmark( scINVALID | scREALIGN );

        SCDebugTrace( 2, scString( "\tCOLEndReformat: col 0x%08x %d\n" ), fCol, fCol->GetCount( ) );
    }
}

/* ==================================================================== */
// add all lines that need to be repainted to the repaint rect

scXRect& scColumn::RepaintExtent( scXRect& repaintExtents )
{
    scXRect     lineExtents;
    scTextline* txl;

    repaintExtents.Invalidate();
    for ( txl = GetFirstline(); txl; txl = txl->GetNext() ) {
        if ( txl->Marked( scREPAINT ) ) {
            txl->QueryExtents( lineExtents, 1 );
            if ( lineExtents.Width() == 0 )
                lineExtents.x2  = lineExtents.x1 + 1;
            repaintExtents.Union( lineExtents );
            txl->Marked( scREPAINT );
        }
    }
    Marked( scREPAINT );

    return repaintExtents;
}

/* ==================================================================== */
// set the column's vertical justification attribute

void scColumn::SetVJ( eVertJust attr )
{
    if ( GetVertJust() != attr )
        Mark( scREALIGN );
    SetVertJust( attr );
}

/* ==================================================================== */
// this does the actual space allocation within the column

Bool scColumn::GetStrip2( scLINERefData&    lineData,
                          int               breakType,
                          scCOLRefData&     colRefData )
{
    scXRect         tryRect;
    MicroPoint      tryX,
                    tryY,
                    colWidth    = Width( ),
                    colDepth    = Depth(),
                    firstLinePosition;
            InsetHRgn( fRgnH, scCachedStyle::GetCurrentCache().GetRunAroundBorder(), scCachedStyle::
GetCurrentCache().GetRunAroundBorder(), true );

            fRgn = (HRgn*)MEMLockHnd( fRgnH );

            break;

        case eRgnShape:
            scCachedStyle::GetCurrentCache().SetRunAroundBorder( CSrunaroundBorder( fCol->GetAPPName
(), scCachedStyle::GetCurrentCache().GetSpec() ) );

            try {
                fRgnH = NewHRgn( RGNSliverSize( fCol->GetRgn() ) );

                CopyHRgn( fRgnH, fCol->GetRgn() );
                InsetHRgn( fRgnH, scCachedStyle::GetCurrentCache().GetRunAroundBorder(), scCachedSty
le::GetCurrentCache().GetRunAroundBorder(), true );
            }
            catch ( ... ) {
                DisposeHRgn( fRgnH ), fRgnH = 0;
                throw;
            }

            fRgn = (HRgn *)MEMLockHnd( fRgnH );
            break;
    }


    fPData.fInitialLine.fBaseline    = FIRST_LINE_POSITION;
    fPData.fComposedLine.fBaseline   = FIRST_LINE_POSITION;

    return true;
}

/* ================================================================== */
// close out the data structures when finished line breaking in a column

void scCOLRefData::COLFini( Bool finished )
{
    if ( fCol ) {
        scXRect extents;

        scAssert( fCol->Marked( scLAYACTIVE ) && fCol == GetActive() );
        fCol->Unmark( scLAYACTIVE );

            //  SCDebugTrace( 2, scString( "COLEndReformat %d" ), col->fColumnCount );

        ggcS.theActiveColH  = NULL;

        switch ( fCol->GetShapeType() ) {
            default:
                fCol->VertJustify( );
                break;

            case eVertShape:
            case eRgnShape:
                fCol->VertJustify( );
                if ( fRgnH ) {
                    MEMUnlockHnd( fRgnH );
                    DisposeHRgn( fRgnH ), fRgnH = NULL;
                }
                break;

            case eHorzFlex:
            case eFlexShape:
            case eVertFlex:
                fCol->QueryMargins( extents );
                if ( fCol->GetFlowdir().IsHorizontal() ) {
                    if ( fCol->GetShapeType() & eVertFlex )
                        fCol->SetDepth( extents.y2 );
                    if ( fCol->GetShapeType() & eHorzFlex ) {
                        fCol->SetWidth( extents.x2 );
                        fCol->RepositionLines();
```

```
#if SCDEBUG > 1
    if ( fCol->GetPrev() )
        scAssert( !fCol->GetPrev()->Marked( scLAYACTIVE ) );
    scAssert( fCol && ! fCol->Marked( scLAYACTIVE ) );
#endif

    fCol->Mark( scLAYACTIVE );

    ggcS.theActiveColH  = fCol;

    SetActive( fCol );

    scCachedStyle::SetFlowdir( fCol->GetFlowdir() );

        // set these to defaults
    fSavedPrevEnd.Set( LONG_MIN, FIRST_LINE_POSITION );

        // now check to see if we are starting reformatting in the
        // middle of the column, if we are we should set the prevbaseline up
    prevPara = p->GetPrev();
    if ( prevPara ) {
        scTextline* txl = prevPara->GetLastline( );
        if ( txl && txl->GetColumn() == fCol ) {
            scColumn*           tCol = fCol;
            scLEADRefData   lead;
            MicroPoint      baseline = fSavedPrevEnd.y;
            p->LocateFirstLine( *this, p->SpecAtStart(), tCol, baseline, lead, prevParaData );
            scAssert( tCol == fCol );
        }
    }

        /* If this fails, no problem. COLLineListChanges */
        /* will simply repaint ALL lines.                */
    SaveLineList( );

    SetRegion( 0 );

    switch ( fCol->GetShapeType() ) {
        default:
            break;
        case eFlexShape:
        case eHorzFlex:
            if ( fCol->GetFlowdir().IsVertical() ) {
                scTextline* txl = fCol->GetFirstline();
                if ( txl ) {
                    MicroPoint  position        = txl->GetOrigin().x;
                    TypeSpec    ts              = txl->SpecAtStart( );
                    MicroPoint  firstlinepos    = CSfirstLinePosition( fCol->GetAPPName(), ts );

                    scMuPoint trans( mpInfinity - position - firstlinepos, 0 );
                    fCol->TranslateLines( trans );
                    fCol->SetWidth( mpInfinity );
                }
            }
            break;

        case eVertShape:
            scCachedStyle::GetCurrentCache().SetRunAroundBorder( CSrunaroundBorder( fCol->GetAPPName
(), scCachedStyle::GetCurrentCache().GetSpec() ) );
            startV          = (scVertex *)MEMLockHnd( fCol->GetVertList() );

            try {
                fRgnH = NewHRgn( scSliverSize() );
                PolyHRgn( fRgnH, startV );
            }
            catch ( ... ) {
                DisposeHRgn( fRgnH ), fRgnH = 0;
                MEMUnlockHnd( fCol->GetVertList() );
                throw;
            }

            MEMUnlockHnd( fCol->GetVertList() );
```

```
{
    fCol = col;
    if ( col )
        fPData.SetFlowDir( col->GetFlowdir() );
    else
        fPData.SetFlowDir( scFlowDir( ) );
}

/* ====================================================================== */
// free lines marks as invalid and collect their damaged area

void scCOLRefData::FreeInvalidLines( void )
{
    scTextline* txl;
    scTextline* nextTxl;

    for ( txl = fCol->GetFirstline(); txl; txl = nextTxl ) {
        nextTxl = txl->GetNext();
        if ( txl->Marked( scINVALID ) )
            txl->Delete( fLineDamage );
        else if ( txl->Marked( scREPAINT ) ) {
            scXRect damage;
            txl->QueryExtents( damage, 1 );
            fLineDamage.Union( damage );
            txl->Unmark( scREPAINT );
        }
    }
}

/* ====================================================================== */
// save the linelist for damage determination in formatting

void scCOLRefData::SaveLineList( )
{
    scTextline* txl;

    for ( txl = fCol->GetFirstline(); txl; txl = txl->GetNext() ) {
        if ( txl->Marked( scREPAINT ) ) {
            scXRect xrect;
            txl->QueryExtents( xrect, 1 );
            fLineDamage.Union( xrect );
            txl->Marked( scREPAINT );
        }
    }

    fSavedLineState.SaveLineList( fCol );
}

/* ====================================================================== */
// initialize the the data structures to perform linebreaking in a column,
// this is primarily used for irregular run-arounds

Bool scCOLRefData::COLInit( scColumn* col, scContUnit* p )
{
    fCol = col;
    fLineDamage.Invalidate();

    FreeInvalidLines( );

    SCDebugTrace( 2, scString( "\tCOLStartReformat: col 0x%08x %d\n" ), fCol, fCol->GetCount( ) );
    scVertex*       startV;
    scContUnit*     prevPara;
    PrevParaData    prevParaData;


    prevParaData.lastLineH  = NULL;
    prevParaData.lastSpec.clear();

//  SCDebugTrace( 2, scString( "COLStartReformat %d" ), col->fColumnCount );

    if ( !fCol->DamOpen() )
        return false;
```

```
/**********************************************************************

    File:       SCCOLUM3.C

    $Header: /Projects/Toolbox/ct/Sccolum3.cpp 3     5/30/97 8:45a Wmanis $

    Contains:   Contains the code to allocate lines for containers and
                other miscellaneous code.

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

**********************************************************************/

#include "scpubobj.h"
#include "sccolumn.h"
#include "scstcach.h"
#include "scglobda.h"
#include "sccallbk.h"
#include "scmem.h"
#include "scparagr.h"
#include "scregion.h"
#include "sctextli.h"
#include "screfdat.h"


/* ==================================================================== */
// translate the lines

void scColumn::TranslateLines( const scMuPoint& trans )
{
    scTextline* txl;

    for ( txl = fFirstline; txl; txl = txl->GetNext() ) {
//      scAssert( txl->fOrigin.x + trans.x >= 0 );
        txl->Translate( trans );
    }
}


/* ==================================================================== */
/* reposition or realign the lines in this column which is probably a
 * flex column
 */

void scColumn::RepositionLines( )
{
    scTextline  *txl;
    MicroPoint  measure;

    if ( GetFlowdir().IsHorizontal() )
        measure = Width( );
    else
        measure = Depth();

    for ( txl = fFirstline; txl; txl = txl->GetNext() )
        txl->Reposition( measure );
}

/* ==================================================================== */
// set the column as the active container in the reformatting cache

void scCOLRefData::SetActive( scColumn* col )
```

```
            redispList->AddColumn( col, lineDamage );
        col->Unmark( scREPAINT );

            // free the list
        if ( !fUsingStoredData  )
            delete [] fData;

        fData       = NULL;
        fNumItems   = 0;
    }
}

/*=================================================================*/
```

```
            txlCopy->SetInkExtents( xrect );
#endif
        }
        scAssert( txl == NULL );
    }
    else
         fData  = NULL;
}

/*=============================================================*/
// compare the list of saved lines with the current column lines and
// determine the repainting that needs to be done

void scRedisplayStoredLine::LineListChanges( scColumn*      col,
                                             const scXRect& oldLineDamage,
                                             scRedispList*  redispList )
{
    scTextline* txlOrg;
    scTextline* txl;
    scXRect     lineDamage( oldLineDamage );
    ushort      lines       = col->GetLinecount();

    scStreamChangeInfo streamChange;

    streamChange = gStreamChangeInfo;

    if (  fData  == NULL ) {
            // redraw the entire column
        if ( redispList ) {
            col->QueryMargins( fOrgExtents );
            redispList->AddColumn( col, fOrgExtents );
        }
        col->Unmark( scREPAINT );
    }
    else {
        txl = col->GetFirstline();


            //
            // compare old lines and new lines and where they differ
            // mark that area to be repainted
            //
        for ( txlOrg = fData; lines &&  fNumItems; txl = LNNext( txl ), txlOrg++ ) {
            lines--;
            fNumItems --;
            if ( !txl->Compare( txlOrg, streamChange ) ) {

                    // handle old line position now
                scXRect xrect,
                        xrect2;
                txl->QueryExtents( xrect, 1 );
                txlOrg->QueryExtents( xrect2, 1 );
                lineDamage.Union( xrect );
                lineDamage.Union( xrect2 );
                txl->Unmark( scREPAINT );
            }
        }

            // fData ran out first, mark the rest of the new lines
        for ( ; lines--; txl = LNNext( txl ) ) {
            scXRect xrect;
            txl->QueryExtents( xrect, 1 );
            lineDamage.Union( xrect );
        }

            // current lines ran out first
        for( ;  fNumItems--; txlOrg++ ) {
            scXRect xrect;
            txlOrg->QueryExtents( xrect, 1 );
            lineDamage.Union( xrect );
        }

        if ( redispList )
```

```
scRedisplayStoredLine::~scRedisplayStoredLine(  )
{
    delete [] fStoredData, fStoredData = NULL;
    fStoredLines    = 0;
}

/*==================================================================*/

void scRedisplayStoredLine::LineListInit( int lines )
{
    fUsingStoredData    = false;

    fData               = 0;
    fNumItems           = 0;

#ifndef MWERKS_NEW_ARRAY_PROBLEM
    fStoredData         = SCNEW scTextline[ lines ];
#else
    fStoredData         = new scTextline[ lines ];
#endif

    fStoredLines        = (short)lines;
}

/*==================================================================*/

void scRedisplayStoredLine::LineListFini(  )
{
    ushort i;
    for ( i = 0; i < fStoredLines; i++ )
        fStoredData[i].InitForReuse( 0 );

    delete [] fStoredData, fStoredData = NULL;
    fStoredLines    = 0;
}

/*==================================================================*/
/* save an image of the lines in a column to determine repainting
 * at the completion of reformatting
 */

void scRedisplayStoredLine::SaveLineList( scColumn* col )
{
    scTextline* txlCopy;
    scTextline* txl;
    ushort      lines;

    lines   = col->GetLinecount( );

    fNumItems   = lines;
    fOrgExtents = col->GetInkExtents();

    if ( lines ) {
            // determine if we are using the stored lines ( about 200 )
            // or do we dynamically allocate a list - if more than 200 lines
            // - which should be almost never
            //
        if (  fStoredData && lines < fStoredLines  ) {
            fUsingStoredData    = true;
            fData               = fStoredData;
        }
        else {
            fData  = new scTextline[ lines ];
            fUsingStoredData  = false;
        }

            // copy current state to the list of lines
        txl = col->GetFirstline();
        for ( txlCopy =  fData ; lines--; txl = LNNext( txl ), txlCopy++ ) {
            *txlCopy = *txl;
#if 0
            scXRect xrect;
            txl->QueryExtents( xrect, 1 );
```

```
void scColumn::VertJustify( )
{
    eVertJust       attributes      = GetVertJust();
    eColShapeType   colShape        = GetShapeType();

    if ( ! ( colShape == eNoShape  || ( colShape & eFlexShape )  ) )
        COLFlushTop( this );
    else {
        switch ( attributes ) {
            case eVertJustified:
                if ( !GetNext() ) {
                        /* If this is the stream's last column, don't VJ    */
                        /* unless force VJ is set. If it isn't the last     */
                        /* column, fall through to the next case to VJ.     */
                    COLFlushTop( this );        // remove effects of vj
                    break;
                }
                // let this fall thru

            case eVertForceJustify:
                COLFlushTopBottom( this );
                break;

            case eVertBottom:
                COLFlushBottom( this, eVJBottom );
                break;

            case eVertCentered:
                COLFlushBottom( this, eVJCenter );
                break;

            default:
            case eVertTop:
                COLFlushTop( this );
                break;
        }
    }
}
/*================================================================*/
/* determine the number of lines in a column */

ushort scColumn::GetLinecount( ) const
{
    scTextline* txl;
    ushort      lineCount;

    for ( lineCount = 0, txl = GetFirstline(); txl; txl = txl->GetNext() )
        lineCount++;
    return lineCount;
}

/*================================================================*/
/* The functions that follow are used to keep track of which lines   */
/* move during VJ, to minimize repainting. If any of it fails due    */
/* lack of memory, VJ is not jeopardized, but everything will end    */
/* up being repainted.                                               */

/*==============================================================*/

scRedisplayStoredLine::scRedisplayStoredLine( int lines ) :
                            fStoredData( 0 ),
                            fStoredLines( 0 ),
                            fUsingStoredData( false ),
                            fData( 0 ),
                            fNumItems( 0 )
{
    LineListInit( lines );
}

/*==============================================================*/
```

```
        col->SetInkExtents( 0, 0, 0, 0 );

        for ( txl = col->GetFirstline(); txl; txl = LNNext( txl ) ) {
            lnParaH = txl->GetPara( );

            /* Shift each line down the appropriate amount  */

            if ( paraH == NULL )
                txl->SetVJ( 0 );
            else if ( paraH == txl->GetPara() ) {
                    /* Add line space */
                adjustment += LineShift( lineSpace, lineStretchFactor, lineAdj );
                txl->SetVJ( adjustment );
                lineAdj++;
            }
            else if ( paraH != tLine->GetPara() ) {
                    /* Add para space */
                adjustment += LineShift( paraSpace, paraStretchFactor, paraAdj );
                txl->SetVJ( adjustment );
                paraAdj++;
            }
            paraH = lnParaH;
            txl->QueryExtents( lineRect );
            col->UnionInkExtents( lineRect );
        }
    }
    catch( ... ) {
        MEMFreePtr( lineSpace );
        MEMFreePtr( paraSpace );
        throw;
    }

    MEMFreePtr( lineSpace );
    MEMFreePtr( paraSpace );

}

/*==================================================================*/
/* shove the lines to the top */

static void COLFlushTop( scColumn* col )
{
    for ( scTextline* txl = col->GetFirstline(); txl; txl = LNNext( txl ) )
        txl->RemoveVJ( );
}

/*==================================================================*/

void scColumn::SetDepthNVJ( MicroPoint      dimension,
                            scRedispList*   redispList )
{
    scXRect lineDamage;

    if ( Marked( scINVALID ) )
        LimitDamage( redispList, scReformatTimeSlice );

    if ( fFlowDir.IsHorizontal() )
        SetDepth( dimension );
    else
        SetWidth( dimension );

    scRedisplayStoredLine rdl( GetLinecount() );
    rdl.SaveLineList( this );
    VertJustify( );
    rdl.LineListChanges( this, lineDamage, redispList );

}

/*==================================================================*/
/* align the text lines in the column,
 * this function just serves as a dispatcher
 */
```

```
                /* If we can do it with para spacing alone, */
                /* do it. -- REAL / MicroPoint              */
            paraStretchFactor = 1 + ((REAL)vDiff) / currParaSpace;
        }
        else {
            if ( currParaSpace > 0 ) {
                    /* Start off by stretching paragraph spacing    */
                    /* to the max.                                  */
                paraStretchFactor   = maxParaStretch;
                vDiff               -= maxTotalParaStretch;
            }

            if ( maxTotalLineStretch >= vDiff && currLineSpace > 0 ) {
                    /* If we can do remaining VJ within */
                    /* max line spacing, do it --  REAL / MicroPoint */
                lineStretchFactor = 1 + ((REAL)vDiff) / currLineSpace;
            }
            else {
                if ( currLineSpace > 0 ) {
                        /* Stretch line spacing to the max, */
                        /* and see what's left over         */
                    lineStretchFactor   = maxLineStretch;
                    vDiff               -= maxTotalLineStretch;
                }

                if ( exceedMaxValues ) {
                    if ( currParaSpace > 0 && extraPPspacing ) {
                            /* If extraPPspacing were true (it isn't),  */
                            /* we would simply increase para spacing    */
                            /* to cover the excess.                     */
                        paraStretchFactor = 1 + ( ((REAL)vDiff)
                                    + maxTotalParaStretch ) / currParaSpace;

                    }       /* (REAL + MicroPoint) / MicroPoint         */
                    else {
                        /* Spread remaining space evenly over all remaining
                         * lines, including both inter line and inter para
                         * spacing.
                         */
                        /* Some care is requires to do it evenly.   */
                        MicroPoint  totalParaSpace
                            = currParaSpace ? currParaSpace+maxTotalParaStretch:0;

                        MicroPoint  totalLineSpace
                            = currLineSpace ? currLineSpace+maxTotalLineStretch:0;

                        MicroPoint  totalSpace
                            = totalParaSpace + totalLineSpace;

                        MicroPoint  paraDiff
                            = scRoundMP( ((REAL)totalParaSpace) / totalSpace * vDiff );

                        MicroPoint  lineDiff
                            = scRoundMP( ((REAL)totalLineSpace) / totalSpace * vDiff );

                        /* REAL / MicroPoint / MicroPoint   */

                        if ( currParaSpace )
                            paraStretchFactor   = 1 + ( ((REAL)paraDiff)
                                        + maxTotalParaStretch ) / currParaSpace;
                        if ( currLineSpace )
                            lineStretchFactor   = 1 + ( ((REAL)lineDiff)
                                        + maxTotalLineStretch ) / currLineSpace;
                        /* ( REAL + MicroPoint ) / MicroPoint   */
                    }
                }
            }
        }

        adjustment  = 0;
        lineAdj     = paraAdj   = 0;
        paraH       = NULL;
```

```
	currLineSpace	= 0;	/* These represent the current total line	*/
	currParaSpace	= 0;	/* and para spacing before VJ			*/

	for ( ; txl; txl = LNNext( txl ) ) {
	    MicroPoint maxlead  = txl->MaxLead( spec );
	    tLine   = txl;
	    scCachedStyle& cs = scCachedStyle::GetCachedStyle( spec );

	    if ( paraH == NULL )
	        ;
	    else if ( paraH == tLine->GetPara() ) {
	        lead = cs.GetComputedLead( );
	            /* Accumulate line space information for each line  */
	        InsertSpaceRecord( lineSpace, lead, cs.GetComputedMaxLead(), interLine++ );
	        currLineSpace += lead;
	    }
	    else if ( paraH != tLine->GetPara() ) {
	        lead = scCachedStyle::GetParaSpace( paraH, tLine->GetPara() );
	        maxlead = scCachedStyle::GetMaxParaSpace( paraH, tLine->GetPara() );

	            /* Accumulate para space information for each para  */
	        InsertSpaceRecord( paraSpace, lead, maxlead, interPara++ );
	        currParaSpace += lead;
	    }

	    paraH       = tLine->GetPara();

	    if ( vertical )
	        currDepth    = MIN( tLine->GetOrigin().x, currDepth );
	    else
	        currDepth    = MAX( tLine->GetOrigin().y, currDepth );
	                /* This will tell us the    */
	                /* depth of the last line   */
	}

	/* Calculate the difference between where the last line is and
	 * where we want it to be
	 */
	if ( vertical )
	    vDiff = -CSlastLinePosition( col->GetAPPName(), spec ) + currDepth;
	else
	    vDiff = col->Depth() - CSlastLinePosition( col->GetAPPName(), spec ) - currDepth;

	    /* The greatest factors by which we can     */
	    /* multiply the space of each line and para */
	maxLineStretch      = MaxSpaceStretch( lineSpace, interLine );
	maxParaStretch      = MaxSpaceStretch( paraSpace, interPara );

	    /* How much space this  */
	    /* will buy us          */
	maxTotalLineStretch = TotalSpaceStretch( lineSpace, interLine,
	                                          maxLineStretch );
	maxTotalParaStretch = TotalSpaceStretch( paraSpace, interPara,
	                                          maxParaStretch );

	    /* How much we are currently stretching the line space  */
	    /* and para space                                       */
	lineStretchFactor   = 1;
	paraStretchFactor   = 1;

	    /* If VJ is impossible or unnecessary   */
	if ( currParaSpace < 0
	        || currLineSpace < 0
	        || ( currParaSpace == 0 && currLineSpace == 0 )
	        || maxLineStretch < 0
	        || maxParaStretch < 0
	        || vDiff <= 0 )
	{
	    COLFlushTop( col );
	    return;
	}

	if ( maxTotalParaStretch >= vDiff && currParaSpace > 0 ) {
```

```
                /* the specified distance       */
            txl->SetVJ( vDiff );
            txl->QueryExtents( lineRect );
            col->UnionInkExtents( lineRect );
        }
    }
}


/*=================================================================*/
/* Vertical justification on a column. Includes both feathering and  */
/* paragraph spacing.                                                */

// TOOLBOX BEHAVIOR
//      We will exceed maximum values to VJ at all costs
//      In such excessive conditions, we won't use extra
//      para spacing to achieve our end; we will use
//      extra line spacing.
//              or
//      We will not exceed max values and thus may not achieve
//      vertical justification
//
static const Bool       exceedMaxValues     = false;
static const Bool       extraPPspacing      = false;


static void COLFlushTopBottom( scColumn *col )
{
    VJSpace*    lineSpace = 0;
    VJSpace*    paraSpace = 0;

    scTextline*     txl;
    scTextline*     tLine;
    TypeSpec        spec;
    scContUnit* lnParaH;
    scContUnit* paraH = NULL;
    short           interPara,
                    interLine,
                    lineAdj,
                    paraAdj;
    MicroPoint      vDiff,
                    currDepth    = LONG_MIN,
                    currLineSpace,
                    currParaSpace,
                    maxTotalLineStretch,
                    maxTotalParaStretch,
                    adjustment,
                    lead;
    REAL            maxLineStretch,
                    maxParaStretch,
                    lineStretchFactor,
                    paraStretchFactor;
    scXRect         lineRect;
    Bool            vertical    = false;

    COLFlushTop( col );         // remove effects of vj

    if ( col->GetFlowdir().IsVertical() ) {
        vertical    = true;
        currDepth   = LONG_MAX;
    }

    /* These handles will store arrays of structures to represent   */
    /* the optimum and the maximum spacing for each line in         */
    /* the column, and for each paragraph in the column.            */

    try {
        lineSpace = (VJSpace*)MEMAllocPtr( sizeof( VJSpace ) * growUnits );
        paraSpace = (VJSpace*)MEMAllocPtr( sizeof( VJSpace ) * growUnits );

        interPara   = interLine = 0;
        txl     = col->GetFirstline();
    scTextline  *tLine;
    TypeSpec    spec;
    MicroPoint  vDiff,
                capHeight,
                maxDepth    = LONG_MIN;
    scXRect     lineRect;
    RLU         capHiteRlu, d1, d2, d3;
    scRLURect   rect;
    Bool        vertical    = false;

    COLFlushTop( col );   // remove effects of previous vj

    if ( col->GetFlowdir().IsVertical() ) {
        vertical = true;
        maxDepth = LONG_MAX;
    }

    lastLineH   = txl   = col->GetFirstline();
    if ( txl == NULL ) {
        return;
    }

    for ( ; txl != NULL; txl = LNNext( txl ) ) {
        tLine       = txl;
        if ( vertical )
            maxDepth    = MIN( tLine->GetOrigin().x, maxDepth );
        else {
                /* Find the depth of the last line */
            maxDepth    = MAX( tLine->GetOrigin().y, maxDepth );
        }
        lastLineH   = txl;
    }

        /* Calculate the distance between the last possible */
        /* line position and our last actual position       */
    if ( vertical )
        vDiff   = maxDepth;
    else
        vDiff   = col->Depth() - maxDepth;

    MicroPoint maxlead  = lastLineH->MaxLead( spec );
    vDiff               -= CSlastLinePosition( col->GetAPPName(), spec );

    if ( flag == eVJCenter ) {
        /* For center justification, cut the distance to move each
         * line in half Further adjustment must be made for the vertical
         * space occupied by the first line.
         */
        if ( ( txl = col->GetFirstline() ) != NULL ) {

            MicroPoint maxlead = txl->MaxLead( spec );

            scCachedStyle::SetFlowdir( col->GetFlowdir() );
            scCachedStyle::GetCachedStyle( spec );
            FIgetRLUFontExtents( scCachedStyle::GetCurrentCache().GetSpec(), capHiteRlu, d1, d2, d3,
 rect );

            capHeight = scRoundMP( (REAL)scCachedStyle::GetCurrentCache().GetPtSize() / scBaseRLUsys
tem * capHiteRlu );

            vDiff -= CSfirstLinePosition( col->GetAPPName(), spec );
            vDiff += capHeight;

            vDiff /= 2;
        }
    }

    if ( vDiff != 0 ) {

        col->SetInkExtents( 0, 0, 0, 0 );

        for ( txl = col->GetFirstline(); txl != NULL; txl = LNNext( txl ) ) {
                /* Shift all the lines down by  */
    VJSpace*    spacePtr = space;
    spacePtr += numRecords;

    spacePtr->opt               = opt;
    spacePtr->max               = max;
    spacePtr->upperBound        = (REAL)max / opt;
}

/*=================================================================*/
/* Return the minimum line space multiplier allowed                */
/* by any line in the column.                                      */

static REAL MaxSpaceStretch( VJSpace*   space,
                             size_t     numRecords )
{
    REAL            maxStretch;

    if ( numRecords-- ) {
        maxStretch = space->upperBound;
        space++;
    }
    else
        return 0;

    for ( ; numRecords--; space++ ) {
        if ( space->upperBound < maxStretch )
            maxStretch = space->upperBound;
    }

    return maxStretch;
}

/*=================================================================*/
/* Return the total line space expansion if every line's space is  */
/* multiplied by maxStretch.                                       */

static MicroPoint TotalSpaceStretch( VJSpace*   space,
                                     size_t     numRecords,
                                     REAL       maxStretch )
{
    REAL            totalStretch = 0;

    for ( ; numRecords--; space++ )
        totalStretch += space->opt * ( maxStretch - 1 );

    return scRoundMP( totalStretch );
}

/*=================================================================*/
/* Return the product of the optimum line space and                */
/* the stretchFactor to calculate the distance to shift a line down. */

static MicroPoint LineShift( VJSpace*   space,
                             REAL       stretchFactor,
                             short      index )
{
    MicroPoint      shift;

        /* MicroPoint * REAL */
    shift = (MicroPoint)( space[index].opt * ( stretchFactor - 1 ) );

    return shift;
}

/*=================================================================*/
/* If flag == center, we do center vertical justification          */
/* Otherwise, we do flush bottom.                                  */

static void COLFlushBottom( scColumn*   col,
                            eDoVJ       flag )
{
    scTextline  *txl;
    scTextline  *lastLineH;
```

```
/****************************************************************************

    File:       SCCOLUM2.C


    $Header: /Projects/Toolbox/ct/SCCOLUM2.CPP 2     5/30/97 8:45a Wmanis $

    Contains:   The code to vj columns and to save the line state
                before reformatting and then compare it post reformatting
                to determine redisplay.

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

****************************************************************************/

#include "sccolumn.h"
#include "scglobda.h"
#include "sctextli.h"
#include "scpubobj.h"
#include "scmem.h"
#include "scexcept.h"
#include "screfdat.h"
#include "sccallbk.h"
#include "scstcach.h"
#include "scstream.h"
#include <limits.h>

/*************************************************************************/
/*************************************************************************/

struct VJSpace {
    MicroPoint  opt;
    MicroPoint  max;
    REAL        upperBound;
};


/*=====================================================================*/

static void     COLFlushTop( scColumn* );
static void     COLFlushTopBottom( scColumn* );

enum eDoVJ {
    eVJBottom = 1,
    eVJCenter
};

#define     growUnits   30
#define     growSize    ( sizeof( VJSpace ) * growUnits )

/*=====================================================================*/
/* Add a space record to the end of the handle.                        */

static void InsertSpaceRecord( VJSpace*&    space,
                               MicroPoint   opt,
                               MicroPoint   max,
                               size_t       numRecords )
{
    size_t newSize  = ( numRecords + 1 ) * sizeof( VJSpace );

    if ( MEMGetSizePtr( space ) <= newSize * sizeof( VJSpace ) )
        space = (VJSpace*)MEMResizePtr( (void**)&space, newSize * sizeof( VJSpace ) );
    Unlock();
}

/* ===================================================================== */

status scRedispList::CL_GetColumnData( APPColumn        appname,
                                       scColRedisplay&  data ) const
{
    status          stat    = scSuccess;
    volatile int    locked  = false;
    volatile int    found   = false;

    try {
        long            limit       = GetNumItems();
        scColRedisplay* colredisp   = (scColRedisplay*)Lock();
        locked                      = true;

        for ( ; limit--; colredisp++ ) {
            if ( colredisp->fAPPName == appname ) {
                data = *colredisp;
                found = true;
            }
        }
        raise_if( found == false, scERRstructure );
    }
    IGNORE_RERAISE;

    return stat;
}

/* ===================================================================== */
```

09620370 .073400

```
        Unlock();
        AddCell( colRefData.fCol );
        Lock();
        cell = FindCell( colRefData.fCol );
    }

    colRefData.fCol->ComputeInkExtents( );
    cell->fWidth                = colRefData.fCol->Width();
    cell->fDepth                = colRefData.fCol->Depth();
    cell->fExRect               = colRefData.fCol->GetInkExtents();
    cell->fAdditionalText       = colRefData.fCol->MoreText( );

    scXRect fRepaintRect( cell->fRepaintRect );
    fRepaintRect.Union(  colRefData.fLineDamage );
    cell->fRepaintRect          = fRepaintRect;
    cell->fHasRepaint           = fRepaintRect.Valid();

    scXRect fDamageRect( cell->fDamageRect );
    fDamageRect.Union(  colRefData.fLineDamage );
    cell->fDamageRect           = fDamageRect;
    cell->fHasDamage            = fDamageRect.Valid();


    Unlock();
}

/* ==================================================================== */

void scRedispList::AddColumn( scColumn* col, scXRect& xrect )
{
    Lock();

    scColRedisplay* cell = FindCell( col );

    if ( !cell ) {
        Unlock();
        AddCell( col );
        Lock();
        cell = FindCell( col );
    }

    col->ComputeInkExtents( );
    cell->fWidth                = col->Width();
    cell->fDepth                = col->Depth();
    cell->fExRect               = col->GetInkExtents();
    cell->fAdditionalText       = col->MoreText();

    scXRect fRepaintRect( cell->fRepaintRect );
    fRepaintRect.Union( xrect );
    cell->fRepaintRect    = fRepaintRect;
    cell->fHasRepaint     = fRepaintRect.Valid();

    Unlock();
}

/* ==================================================================== */

void scRedispList::SetImmediateRect( scColumn*                  col,
                                     const scImmediateRedisp&   immedredisp )
{
    Lock();

    scColRedisplay* cell = FindCell( col );

    if ( !cell ) {
        Unlock();
        AddCell( col );
        Lock();
        cell = FindCell( col );
    }

    cell->fImmediateRedisplay   = true;
    cell->fImmediateArea        = immedredisp;
```

```
/*****************************************************************************

    File:       SCCOLINF.C


    $Header: /Projects/Toolbox/ct/SCCOLINF.CPP 2     5/30/97 8:45a Wmanis $

    Contains:   code to collect column redisplay information

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox Application software is the proprietary
    and confidential property of Stonehand Inc.

*****************************************************************************/

#include "scpubobj.h"
#include "sccolumn.h"
#include "scglobda.h"
#include "screfdat.h"

/* ======================================================================== */

void scRedispList::ReInit( )
{
}

/* ======================================================================== */

scColRedisplay* scRedispList::FindCell( const scColumn* col ) const
{
    long            limit       = GetNumItems();
    scColRedisplay* colredisp   = (scColRedisplay*)Lock();

    for ( ; limit--; colredisp++ ) {
        if ( colredisp->fColumnID == col ) {
            Unlock();
            return colredisp;
        }
    }

    Unlock();

    return 0;
}

/* ======================================================================== */

void scRedispList::AddCell( scColumn* col )
{
    scAssert( !FindCell( col ) );

    scColRedisplay colredisp( col, col->GetAPPName() );

    AppendData( (ElementPtr)&colredisp );
}

/* ======================================================================== */

void scRedispList::AddColumn( const scCOLRefData& colRefData )
{
    Lock();

    scColRedisplay* cell = FindCell( colRefData.fCol );

    if ( !cell ) {
    };
};

/* ==================================================================== */


inline scCharFlags& MkChrFlgs( ushort& flags )
{
    return *(scCharFlags*)&flags;
}

#endif /* _H_SCCHAR */
```

```
                    return f2_.warichu_;
                }

    Bool        IsSpecialNihon( void ) const
                {
                    return f2_.renmoji_ || f2_.rubi_ || f2_.warichu_;
                }

    void        ClrSpecialNihon( void )
                {
                    ClrRubi();
                    ClrRenMoji();
                    ClrWarichu();
                }

    void        SetSpacePosition( unsigned val )
                {
                    f2_.spacepos_ = val;
                }
    void        ClrSpacePosition( void )
                {
                    f2_.spacepos_ = 0;
                }
    unsigned    GetSpacePosition( void ) const
                {
                    return f2_.spacepos_;
                }

    void        SetField( uint8 field )
                {
                    f1_.fField = field;
                }
    uint8       GetField( ) const
                {
                    return (uint8)f1_.fField;
                }

    Bool        IsBreakable( void ) const
                {
                    return !( f2_.renmoji_ || f2_.rubi_ || f2_.warichu_ || f1_.fNoBreak );
                }

private:
    void        ClearMinFlags( void )
                {
                    f1_.fDiscHyph   = 0;
                    f1_.fNoBreak        = 0;
                    f1_.fHyphLevel  = 0;
                    f1_.fAutoKern   = 0;
                    f1_.fDropCap    = 0;
                    f1_.fLineBreak  = 0;
                }
    void        ClearAllFlags( void )
                {
                    f2_.dischyph_   = 0;
                    f2_.nobreak_    = 0;
                    f2_.hyphlevel_  = 0;
                    f2_.autokern_   = 0;
                    f2_.dropcap_    = 0;
                    f2_.linebreak_  = 0;

                    f2_.spacepos_   = 0;
                    f2_.warichu_    = 0;
                    f2_.rubi_       = 0;
                    f2_.renmoji_    = 0;
                }

    union {
        scCharFlags1    f1_;
        scCharFlags2    f2_;
        uint32          f__;
```

```
                {
                    f1_.fDiscHyph = 1;
                }
    void        ClrDiscHyphen( void )
                {
                    f1_.fDiscHyph = 0;
                }
    Bool        IsDiscHyphen( void ) const
                {
                    return f1_.fDiscHyph;
                }

    void        SetNoBreak( void )
                {
                    f1_.fNoBreak = 1;
                }
    void        ClrNoBreak( void )
                {
                    f1_.fNoBreak = 0;
                }
    Bool        IsNoBreak( void ) const
                {
                    return f1_.fNoBreak;
                }

    Bool        IsHyphPresent( void ) const
                {
                    return GetHyphLevel()||IsDiscHyphen();
                }

    void        ClrAutoBits( void )
                {
                    ClrAutoHyphen();
                    ClrKernBits();
                }

    void        SetRubi( void )
                {
                    f2_.rubi_ = 1;
                }
    void        ClrRubi( void )
                {
                    f2_.rubi_ = 0;
                }
    Bool        IsRubi( void ) const
                {
                    return f2_.rubi_;
                }

    void        SetRenMoji( unsigned val )
                {
                    f2_.renmoji_ = val;
                }
    void        ClrRenMoji( void )
                {
                    f2_.renmoji_ = 0;
                }
    unsigned    GetRenMoji( void ) const
                {
                    return f2_.renmoji_;
                }

    void        SetWarichu( unsigned val )
                {
                    f2_.warichu_ = val;
                }
    void        ClrWarichu( void )
                {
                    f2_.warichu_ = 0;
                }
    unsigned    GetWarichu( void ) const
                {
```

```
    void        ClrVarious( void )
                {
                    f1_.fLineBreak = 0;
                    f1_.fHyphLevel = 0;
                }



    int         operator==( const scCharFlags& flags ) const
                {
                    return f__ == flags.f__;
                }

//  scCharFlags&    operator=( const scCharFlags& flags )
//              {
//                  f__ = flags.f__;
//                  return *this;
//              }

    void        SetLineBreak(void)
                {
                    f1_.fLineBreak = 1;
                }
    void        ClrLineBreak(void)
                {
                    f1_.fLineBreak = 0;
                }
    Bool        IsLineBreak(void) const
                {
                    return f1_.fLineBreak;
                }

    void        SetDropCap( void )
                {
                    f1_.fDropCap = 1;
                }
    void        ClrDropCap( void )
                {
                    f1_.fDropCap = 0;
                }
    Bool        IsDropCap( void ) const
                {
                    return f1_.fDropCap;
                }

    void        SetKernBits( void )
                {
                    f1_.fAutoKern = 1;
                }
    void        ClrKernBits( void )
                {
                    f1_.fAutoKern = 0;
                }
    Bool        IsKernPresent( void ) const
                {
                    return f1_.fAutoKern;
                }

    void        SetAutoHyphen( unsigned val )
                {
                    f1_.fHyphLevel = val;
                }
    void        ClrAutoHyphen( void )
                {
                    f1_.fHyphLevel = 0;
                }
    unsigned    GetHyphLevel( void ) const
                {
                    return f1_.fHyphLevel;
                }

    void        SetDiscHyphen( void )
```

```
/***************************************************************************

    File:       SCCHAR.H


    $Header: /Projects/Toolbox/ct/SCCHFLAG.H 2     5/30/97 8:45a Wmanis $

    Contains:   Flags for the glyph processing.

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

***************************************************************************/

#ifndef _H_SCCHAR
#define _H_SCCHAR

#define SIZE_OF_MACHINE     256

/* character definitions */
#define MIN_CHARACTER       START_STREAM
#define MAX_CHARACTER       (SIZE_OF_MACHINE - 1)


struct scCharFlags1 {
    unsigned    fFauxChar   : 16;           // for alignment purposes
    unsigned    fDiscHyph   : 1;
    unsigned    fNoBreak    : 1;
    unsigned    fHyphLevel  : 3;
    unsigned    fAutoKern   : 1;
    unsigned    fDropCap    : 1;        // why do i need this
    unsigned    fLineBreak  : 1;        // why do i need this

    unsigned    fField      : 8;
};


struct scCharFlags2 {
    unsigned    fauxchar_   : 16;           // for alignment purposes
    unsigned    dischyph_   : 1;
    unsigned    nobreak_    : 1;
    unsigned    hyphlevel_  : 3;
    unsigned    autokern_   : 1;
    unsigned    dropcap_    : 1;        // why do i need this
    unsigned    linebreak_  : 1;        // why do i need this

    unsigned    spacepos_   : 2;        // position of space leading or trailing or none in escapeme
nt
    unsigned    warichu_    : 2;        // if non-zero represent # lines
    unsigned    rubi_       : 1;        // annotated character(s)
    unsigned    renmoji_    : 3;        // max target of 7 characters
};


class scCharFlags {
    friend class CharRecord;

public:

    void        ClrCJKVarious( void )
                {
                    ClrVarious();
                    f2_.spacepos_ = 0;
                }
```

```
#define scWordSpace          0x0020
#define scRomanWordSpace     scWordSpace     // ' ' or 0x20 or 32
#define scKanjiWordSpace     0x8140

#define scBreakingHyphen     '-'
#define scNoBreakSpace       0x00a0  /* part of the mac character set */
#define scEnDash             0x00d0
#define scEmDash             0x00d1


inline Bool IsBreakingCharacter( UCS2 ch )
    { return ch == scBreakingHyphen || ch == scEnDash || ch == scEmDash; }



UCS2    CMinputMap( ushort );   /* from APP to Stonehand -
                                     * on file importing
                                     */
UCS2    CMctToAPP( UCS2 );      /* from Stonehand to APP */
UCS2    CMappToCT( UCS2 );      /* from APP to Stonehand */
int     CMcontent( UCS2 );      /* is keystroke a selection change
                                         * or a real input of content
                                         */

void        CMmakeKeyRecordTwo( scKeyRecord&,
                                UCS2,
                                GlyphSize,
                                TypeSpec,
                                Bool,
                                scStreamLocation& );


#endif /* _H_SCCHAREX */
```

09652370.072400

```
/*****************************************************************

    File:       SCCHAREX.H

    $Header: /Projects/Toolbox/ct/SCCHAREX.H 2     5/30/97 8:45a Wmanis $

    Contains:   character exchange from toolbox to outside world

    Written by: Lucas

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.


    #define scIndentSpace       0x0007 deprecated 3/18/96 wam

 ***********************************************************************/

#ifndef _H_SCCHAREX
#define _H_SCCHAREX

#include "sctypes.h"

#define scLeftArrow     ((UCS2)1)
#define scRightArrow    ((UCS2)2)
#define scUpArrow       ((UCS2)3)
#define scDownArrow     ((UCS2)4)
#define scParaSplit     ((UCS2)5)
#define scBackSpace     ((UCS2)6)          // delete character backward
#define scForwardDelete ((UCS2)7)       // delete character forward

// the following are characters actually stored in the stream and do have
// real character codes
#define scEndStream         0x0000
/* 0x0001 is taken */
/* 0x0002 is taken */
/* 0x0003 is taken */
/* 0x0004 is taken */
/* 0x0005 is taken */
/* 0x0006 is taken */
#define scEmptySpace        0x0008  /* a horizontal move that is meaningless to the user */
#define scTabSpace          0x0009  /* part of the mac character set */
#define scHardReturn        0x000a  /* part of the mac character set */
#define scVertTab           0x000b
#define scField             0x000c  /* field character */

/* 0x000d is not taken */
/* 0x000e is not taken */
#define scRulePH            0x000f
/* 0x0010 is not taken */
#define scParaStart         0x0011  /* this has no meaning outside of a report to the client of the
cursor position */
#define scParaEnd           0x0012  /* para break */
#define scQuadCenter        0x0013
#define scQuadLeft          0x0014
#define scQuadRight         0x0015
#define scQuadJustify       0x0016
#define scFixAbsSpace       0x0017  /* absolute fixed space */
#define scFixRelSpace       0x0018  /* relative fixed space stored in rlu's */
#define scFillSpace         0x0019
#define scNoBreakHyph       0x001a
#define scDiscHyphen        0x001b
#define scFigureSpace       0x001c
#define scThinSpace         0x001d
#define scEnSpace           0x001e
#define scEmSpace           0x001f
```

```
    virtual void        release() = 0;
    virtual void        content( stUnivString&, APPColumn, TypeSpec& ) = 0;
};


#endif /* _H_SCCALLBK */
```

09629370.073100

```
                                                                      // ink extents of all glyphs in
                                                                      // font.


/* ================================================================== */
/* =================   HYPHENATION SUB-SYSTEM ======================== */
/* ================================================================== */

// @CALLBACK  Initializes the Hyphenation subs-sytem to the indicated langauge.
// Returns true if language properly inited.
//
Bool scIMPL_IMPORT  HYFLanguageInit(
                        APPLanguage lang );         // @parm <t APPLanguage>

// @CALLBACK  Chars are in word, NULL terminated, return hyph values in hyfs, max
// len of either is 64. if word is hyphenated return true.
//
Bool scIMPL_IMPORT  HYFWord(
                        const UCS2* theWord,        // @parm The word.
                        short*      hyphArray );    // @parm The hyphenation array.

/* ================================================================== */
/* ================================================================== */
/* ================== CHAR DRAWING CALLBACKS ======================== */
/* ================================================================== */
/* ================================================================== */


// @CALLBACK  Called before the start of drawing a line.
//
void scIMPL_IMPORT  APPDrawStartLine(
                        APPDrwCtx       drwctx,     // @parm <t APPDrawCtx>
                        MicroPoint      x,          // @parm X origin of line.
                        MicroPoint      y,          // @parm Y origin of line.
                        const scXRect&  inkext );   // @parm Max ink extents of line.

// @CALLBACK  Called n times ( for each style or full buffer ) between a APPDrawStartLine
// and an APPDrawEndLine.
// @xref <f SCCOL_Update>
void scIMPL_IMPORT  APPDrawString(
                        APPDrwCtx           dc, // @parm Pass thru context.
                        const scGlyphArray* ga, // @parm <t scGlyphArray> array.
                        short               num,// @parm Number of glyphs in array.
                        MicroPoint          x,  // @parm X origin of string.
                        MicroPoint          y,  // @parm Y origin of string.
                        const scGlyphInfo&  gi );// @parm <t scGlyphInfo>

// @CALLBACK  Called at the end  of drawing a line.
void scIMPL_IMPORT  APPDrawEndLine(
                        APPDrwCtx dc );     // @parm <t APPDrwCtx> drawing context.


// @CALLBACK Used to draw hiliting rectangles.
void scIMPL_IMPORT  APPDrawRect(
                        const scXRect& xorRect, // @parm <c scXRect> to xor.
                        APPDrwCtx      dc,      // @parm <t APPDrwCtx> drawing context.
                        Bool           sliverCursor );

void scIMPL_IMPORT  APPDrawRule( const scMuPoint&,
                                 const scMuPoint&,
                                 const scGlyphInfo&,
                                 APPDrwCtx );


/* ================================================================== */
/* ================================================================== */
/* ================================================================== */

class clField {
public:
    static  clField&    createField( scStream*, uint8 );

    virtual uint8       id() const = 0;
```

```
RLU scIMPL_IMPORT   FIgetRLUEscapement(
                        const scFontRender& fr,      // @parm <t scFontRender>
                        UCS2  ch );                  // @parm Glyph.

RLU scIMPL_IMPORT   FIgetRLUEscapement( const scFontRender&,
                                        UCS2,
                                        RLU /*suggestedWidth*/ );

// @CALLBACK  Return the kerning value of the glyphs in design coordinates.
//
RLU scIMPL_IMPORT   FIgetRLUKern(
                        const scFontRender& fr,      // @parm <t scFontRender>
                        UCS2    ch1,                 // @parm Glyph one.
                        UCS2    ch2 );               // @parm Glyph two.

// @CALLBACK  Return the glyph ink box in design coordinates
//
scRLURect& scIMPL_IMPORT    FIgetRLUExtents(
                                const scFontRender& fr,     // @parm <t scFontRender>
                                UCS2                ch,     // @parm Glyph one.
                                scRLURect&      inkBox );   // @parm <c scRLURect>

// @CALLBACK  Return the various font metrics in design coordinates
void scIMPL_IMPORT  FIgetRLUFontExtents(
                        const scFontRender&     fontrender, // @parm <t scFontRender>
                        RLU&                    capHite,    // @parm Cap height.
                        RLU&                    xHite,      // @parm Lower case x height.
                        RLU&                    ascenderHite,   // @parm Ascender height.
                        RLU&                    descenderDepth, // @parm Descender height.
                        scRLURect&              maxInkExt );    // @parm <c scRLURect> union of
                                                                // ink extents of all glyphs in
                                                                // font.


        //////////// DEVICE METRICS ///////////////////////////////

// @CALLBACK  Return the escapement of the glyph in device coordinates
// ( transformed into toolbox coordinates ).
GlyphSize scIMPL_IMPORT     FIgetDEVEscapement(
                                const scFontRender& fr,     // @parm <t scFontRender>
                                UCS2  ch );                 // @parm Glyph.


GlyphSize scIMPL_IMPORT     FIgetDEVEscapement( const scFontRender&,
                                                UCS2,
                                                GlyphSize /*suggestedWidth*/ );

// @CALLBACK  Return the kerning value of the glyphs in device coordinates
// ( transformed into toolbox coordinates ).

GlyphSize scIMPL_IMPORT     FIgetDEVKern(
                        const scFontRender& fr,      // @parm <t scFontRender>
                        UCS2    ch1,                 // @parm Glyph one.
                        UCS2    ch2 );               // @parm Glyph two.

// @CALLBACK  Return the glyph ink box in device coordinates
// ( transformed into toolbox coordinates ).
                            //
scXRect& scIMPL_IMPORT      FIgetDEVExtents(
                                const scFontRender& fr,     // @parm <t scFontRender>
                                UCS2                ch,     // @parm Glyph one.
                                scXRect&            inkBox );// @parm <c scXRect>

// @CALLBACK  Return the various font metrics in device coordinates
// ( transformed into toolbox coordinates ).

void scIMPL_IMPORT  FIgetDEVFontExtents(
                        const scFontRender&     fontrender, // @parm <t scFontRender>
                        MicroPoint&             capHite,    // @parm Cap height.
                        MicroPoint&             xHite,      // @parm Lower case x height.
                        MicroPoint&             ascenderHite,   // @parm Ascender height.
                        MicroPoint&             descenderDepth, // @parm Descender height.
                        scXRect&                maxInkExt );    // @parm <c scXRect> union of
```

```
status  scIMPL_IMPORT   TSGetStyle( TypeSpec& ts,       // @parm <t TypeSpec>
                                    scStyle& style );   // @parm <c scStyle>

// @CALLBACK This call back is used to determine positioning of tabs.
// @ex Default value for tab positioning might be. |
//  tabInfo.xPos = ( xPos / defaultTabWidth + 1 ) * defaultTabWidth );
//
status scIMPL_IMPORT    TSTabInfo(
                            TypeSpec&   paraspec,   // paragraph spec
                            TypeSpec&   ts,         // @parm <t TypeSpec>
                            scTabInfo&  tabInfo,    // @parm <t scTabInfo>
                            MicroPoint  xPos,       // @parm X position in column.
                            MicroPoint  yPos,       // @parm Depth in column.
                            long        lineNum );  // @parm Line num in para.

                        // default wordspace
status scIMPL_IMPORT    TSfillCharInfo( TypeSpec&,
                                        UCS2&,
                                        eFCAlignment&,
                                        MicroPoint,
                                        MicroPoint,
                                        long );

                        // default return false
Bool scIMPL_IMPORT  TSdropCap( TypeSpec&,       // para spec
                               TypeSpec&,       // character spec
                               DCPosition&,     // position struct
                               UCS2 );          // dropcap character
//line Bool         TSdropCap( TypeSpec, DCPosition& ) { return false; }


/*================================================================== */
/*================================================================== */
// COLUMN SPECIFICATIONS - 'CS'

    // By sending in the two specs the spec management system may generate
    // a value intelligently, either a hard coded value or parametrically
    // derived value using the pointsize of the type

// @CALLBACK  Position of first line in a column, default should
// be point size, this is not for use with dropcaps. Client
// may return any reasonable value and may use none, one or
// both of the parameters.
MicroPoint scIMPL_IMPORT    CSfirstLinePosition(
                                APPColumn appcol,       // @parm <t APPColumn>
                                TypeSpec  ts );         // @parm <t TypeSpec>

// @CALLBACK  Position of last line in a column,
// default should be zero since this will allow
// multiple columns with different pointsizes
// to bottom align.
MicroPoint scIMPL_IMPORT    CSlastLinePosition(
                        APPColumn appcol,       // @parm <t APPColumn>
                        TypeSpec  ts );         // @parm <t TypeSpec>

// @CALLBACK  Border to inset text from shape applied to
// column -- default is 0, the spec is the first
// encountered in the column.
inline MicroPoint   CSrunaroundBorder(
                        APPColumn appcol,       // @parm <t APPColumn>
                        TypeSpec  ts )          // @parm <t TypeSpec>
                            { return 0; }

/* ================================================================== */
/* ================ FONT METRIC CALL BACKS ========================== */
/* ================================================================== */

        /////////// DESIGN METRICS //////////////////////////////
        // DESIGN COORDINATES ARE THE RELATIVE UNIT SYSTEM DEFINED
        // IN scBaseRLUSystem

// @CALLBACK  Return the escapement of the glyph in design coordinates.
//
```

```
    scIndRightBL,        scIndentExtra1, scIndentExtra2,
    scNoHyphLastWord,
    scColNoBreak,
    scKeepNext,
    scLinesBefore,
    scLinesAfter,        scWidowOrphanExtra1,scWidowOrphanExtra2,

    scRag,
    scForceJust,
    scRagPattern,
    scRagZone,
    scKernMargins,
    scHLeft,
    scHRight,
    scHLeftAmount,
    scHRightAmount,      scRagExtra1, scRagExtra2,scHPuncExtra1,scHPuncExtra2,

    scHyphenation,
    scHyphChar,
    scHyphLines,
    scHyphExcep,
    scHyphMinSize,
    scPreHyphs,
    scPostHyphs,
    scHyphPropensity,
    scHyphCaps,
    scHyphAcros,         scHyphExtra1, scHyphExtra2,

    scDCShow,
    scDCChar,
    scDCptSize,
    scDCsetSize,
    scDChOffset,
    scDCvOffset,
    scDChBorder,
    scDCvBorder,
    scDCfont,
    scDCcolor,

    scMaxFillChars,
    scFillPos,
    scFillChar,
    scFillAlign,

    scMaxTabs,
    scTabPos,
    scTabAlign,
    scTabChar,
    scTabFillAlign,

    scMinMeasure,
    scRunAroundBorder,
    scFirstLine,

    scMaxValType
} eSpecChange;


// @CALLBACK Used to determine minimal work on a spec change.
// @rdesc <t eSpecTask>
eSpecTask           SpecTaskCalculate(
                         eSpecChange specChange );  // @parm <t eSpecChange>

/* =================================================================== */
/* =================================================================== */
/* ================== SPEC SUB-YSTEM CALL BACKS ===================== */
/* =================================================================== */
/* =================================================================== */

class scStyle;

// @CALLBACK  Gets the scStyle structure.
```

```
void* scIMPL_IMPORT     APPDiskIDToPointer(
                            APPCtxPtr,
                            long    diskID,     // @parm A value returned by <f APPPointerToDiskID>
                                                // that we want a valid pointer to now.
                            stDiskidClass );    // class of object


/* ==================================================================== */
//
// called periodically by the Toolbox during actions
// that will take some time. If the call for an event returns 0,
// the action will be aborted and control will revert to application.
// The client can give the Toolbox and hint as to how much more time
// it can process for. The client can return a negative number as
// an indicator to get out fast.
//

    // this describes the current process type that the toolbox
    // is performing.
typedef enum scProcTypes {
    scDrawProc,     // toolbox is drawing
    scReformatProc  // toolbox is reformatting
} scProcType;


scTicks scIMPL_IMPORT   APPEventAvail( scProcType );

/* ==================================================================== */

// @enum eSpecChange | When a TypeSpec is changed externally to the
// Toolbox, the Toolbox needs to be informed that a change has occurred
// so that reformatting may occur. In an effort to minimize the work
// the function <f SpecTaskCalculate> can calculate the minimum amount
// of work that needs to be done. (e.g. changing the color of a spec
// should only require repainting and not reformattion,
// SpecTaskCalculate(scColor) would return eSCRepaint ) With the return
// value of SpecTaskCalculate one can inform the Toolbox about the changed
// spec <f SCENG_ChangedTS>( ts, <t eSpecTask>, <c scRedispList> ) and
// get information about the minimal area to update.

typedef enum eSpecChanges {
    scLanguage,
    scFont,
    scColor,
    scRenderAttribute,
    scCharTransform,

    scPointSize,
    scSetSize,
    scHoblique,
    scVoblique,
    scRotation,

    scKern,
    scMarginKern,
    scTrack,
    scMinLsp,
    scOptLsp,
    scMaxLsp,

    scMinWsp,
    scOptWsp,
    scMaxWsp,

    scLead,
    scBaseline,
    scAboveLead,
    scBelowLead,

    scIndLines,
    scIndAmount,
    scIndDepth,
    scIndLeftBL,
/****************************************************************************

    File:       sccallbk.h


    $Header: /Projects/Toolbox/ct/SCCALLBK.H 2     5/30/97 8:45a Wmanis $

    Contains:   The call backs to the client from the composition toolbox.

    Written by: Manis

    Copyright (c) 1989-1994 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.
@doc

****************************************************************************/

#ifndef _H_SCSPECSY
#define _H_SCSPECSY

#ifdef SCMACINTOSH
#pragma once
#endif

#include "sctypes.h"

/*======================================================================*/
// The following are call backs that the application must support
// in order for the above selection messages to work properly.
//


// @CALLBACK Provides the Toolbox with the drawing
// context of the column, used for highlighting or drawing.
//
status scIMPL_IMPORT    APPDrawContext(
                            APPColumn       appCol,     // @parm <t APPColumn>
                            const scColumn* col,        // @parm <c scColumn>
                            APPDrwCtx&      drwctx );   // @parm <t APPDrwCtx>


// CALL BACK - this informs the composition toolbox whether it should recompose
// this column or not, the client may prevent recomposition of columns that are not
// visible, though keep in mind that if a subsequent column is visble we
// will have to recompose this column at some point in time.
Bool scIMPL_IMPORT      APPRecomposeColumn( APPColumn );

/* ================================================================== */
// @CALLBACK Maps a pointer of a client object to an ID on disk. Typically
// a TypeSpec.

enum stDiskidClass {
    diskidUnknown,
    diskidColumn,
    diskidTypespec,
    diskidOther
};

long scIMPL_IMPORT      APPPointerToDiskID(
                            APPCtxPtr,
                            void*  clientObj,       // @parm Pointer to client object.
                            stDiskidClass );        // class of object


// @CALLBACK  Maps a disk ID to a pointer. Typically a TypeSpec.
```

```
eBreakType  BRKJapanLineBreak( CharRecordP,
                              long,
                              long&,
                              scLINERefData&,
#ifdef scUseRubi
                              scRubiArray *,
#endif
                              short,
                              short&,
                              scSpecRecord **,
                              scXRect&,
                              GlyphSize&,
                              DCState& );

//MicroPoint BRKComposeRenMoji(  CharRecordP chRec, TypeSpec ts, scFlowDir& fd, Bool fit );

#endif

Bool        BRKJustify( CharRecordP, long, long, MicroPoint );


#endif /* _H_SCBREAK */
```

09620370 .073400

```
/****************************************************************************

    File:       SCBREAK.H


    $Header: /Projects/Toolbox/ct/SCBREAK.H 2     5/30/97 8:45a Wmanis $

    Contains:   LineBreaker interface

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

****************************************************************************/

#ifndef _H_SCBREAK
#define _H_SCBREAK

#ifdef SCMACINTOSH
#pragma once
#endif

#include "sccolumn.h"

typedef enum eHyphenRanks {
    eDiscHyphRank   = 1,
    eBestHyphRank,
    eGoodHyphRank,
    eBadHyphRank
} eHyphenRank;


struct Hyphen {
    short       offset;
    eHyphenRank rank;
};


/* ============================================== */
/* ============================================== */

#include "screfdat.h"

class scSpecRecord;
class scRubiArray;
class scLEADRefData;
class scLINERefData;




class DCState;

eBreakType  BRKRomanLineBreak( CharRecordP,
                               long,
                               long&,
                               scLINERefData&,
                               short,
                               short&,
                               scSpecRecord **,
                               scXRect&,
                               GlyphSize& );

#ifdef scJIS4051
```

```
    return *this;
}

/**************************************************************************/
```

09523970 .073400

```
    else
        noStartline = false;

    if ( cb.fTheBits.fCharClass )
        noEndline   = true;
    else
        noEndline   = false;


    // ValidateBits( theCharacter, cb );

    if ( gbrS.numTargetChars > 0 ) {    /* inhibit breaks in    */
        gbrS.numTargetChars--;              /* target sequence      */
    }
    else {
            /* set a potential break before every character */
        if ( !( noStartline || gbrS.fNoStartline ) ) {
            BRKSetCandBreak( eCharBreak );

            if ( BRKExceedVals( adjustableSpace ) ) {
                BRKLineDecision( 0 );
                return BRKExitLoop( );
            }
        }

        if ( gbrS.firstBox )
            BRKSetFirstBox( );
    }

    gbrS.cB.curBox += gbrS.cB.theChRec->escapement;

    if ( noEndline || ( !scCachedStyle::GetCurrentCache().fmBreakableNumbers && cb.fTheBits.fDigit )
 )
        gbrS.fNoStartline   = true;
    else
        gbrS.fNoStartline   = false;

    if ( cb.fTheBits.fHangable )
        gbrS.cB.fHangable = gbrS.cB.theChRec->escapement;
    else
        gbrS.cB.fHangable = 0;

    gbrS.cB.chCount++;
    gbrS.cB.streamCount++;
    gbrS.cB.theChRec++;
}
#endif

/************************************************************************/

CandBreak& CandBreak::operator=( const CandBreak& cb )
{
    breakCount      = cb.breakCount;
    startCount      = cb.startCount;
    streamCount     = cb.streamCount;
    wsSpaceCount    = cb.wsSpaceCount;
    spaceCount      = cb.spaceCount;
    trailingSpaces  = cb.trailingSpaces;
    chCount         = cb.chCount;
    fillSpCount     = cb.fillSpCount;
    lineVal         = cb.lineVal;
    breakVal        = cb.breakVal;
    minGlue         = cb.minGlue;
    optGlue         = cb.optGlue;
    maxGlue         = cb.maxGlue;
    curBox          = cb.curBox;
    fHangable       = cb.fHangable;
    theChRec        = cb.theChRec;
    specChanged     = cb.specChanged;
    spec            = cb.spec;
    specRec         = cb.specRec;
```

```
/***********************************************************************/

Bool BRKJustify( CharRecordP    chRec,      /* the character array */
                 long   start,      /* count into ch array to start the linebreak */
                 long   stop,       /* count into ch array of end of line */
                 MicroPoint measure )   /* measure to justify to */
{
    long        spaces,
                count;
    MicroPoint  delta;
    MicroPoint  boxWidth;
    CharRecordP holdChRec;
    Bool        changed = false;

    chRec       += start;
    holdChRec   = chRec;

    boxWidth = 0;
    for ( spaces = 0, count = stop - start; count; chRec++, count-- ) {
        switch ( chRec->character ) {
            case scWordSpace:
                if ( BRKStillMoreChars( chRec, (long)count ) )
                    spaces++;
                break;
            default:
                boxWidth += chRec->escapement;
                break;
        }
    }

    delta = measure - boxWidth;
    if ( spaces ) {
        delta = scRoundMP((REAL)delta / spaces);

        for ( chRec=holdChRec, count = stop-start; count; chRec++,count-- ) {
            switch ( chRec->character ) {
                case scWordSpace:
                    if ( spaces ) {
                        spaces--;
                        if ( !changed && chRec->escapement != delta )
                            changed = true;
                        chRec->escapement = (GlyphSize)delta;
                    }
                    break;
                default:
                    break;
            }
        }
    }
    return changed;
}

/***********************************************************************/

#if 0
static void BRKCharJapanese( )
{
    MicroPoint      adjustableSpace;
    UCS2        theCharacter;
    CharBits        cb;
    Bool            noStartline,
                    noEndline;


    adjustableSpace = gbrS.desiredMeasure - gbrS.cB.curBox;
    theCharacter    = gbrS.cB.theChRec->character;

    cb = TSCharBits( scCachedStyle::GetCurrentCache().fmTheSpec, theCharacter );

    if ( cb.fTheBits.fCharClass )
        noStartline = true;
```

```
{
    int i;

    gbrS.breakMach      = new BrFunc[ SIZE_OF_MACHINE ];
    gbrS.fMaxLineVals   = new scMaxLineVals[ MAXLEADVALS ];
    gbrS.candBreak      = new CandBreak[ MAXBREAKVALS ];

    for ( i = 0; i < SIZE_OF_MACHINE; i++ ) {
        switch ( i ) {
            case scTabSpace:
                gbrS.breakMach[i] = bmBRKTab;
                break;
            case scWordSpace:
                gbrS.breakMach[i] = bmBRKWordSpace;
                break;
            case scEndStream:
                gbrS.breakMach[i] = bmBRKEndStream;
                break;
            case scEnDash:
            case scEmDash:
            case scBreakingHyphen:
            case '=':
                gbrS.breakMach[i] = bmBRKHyphen;
                break;
            case scFillSpace:
                gbrS.breakMach[i] = bmBRKFillSpace;
                break;
            case scRulePH:
                gbrS.breakMach[i] = bmBRKRule;
                break;
            case scFixAbsSpace:
            case scFigureSpace:
            case scThinSpace:
            case scEnSpace:
            case scEmSpace:
                gbrS.breakMach[i] = bmBRKFixSpace;
                break;
            case scFixRelSpace:
                gbrS.breakMach[i] = bmBRKRelSpace;
                break;
            case scVertTab:
                gbrS.breakMach[i] = bmBRKVertTab;
                break;
            case scQuadCenter:
            case scQuadLeft:
            case scQuadRight:
            case scQuadJustify:
                gbrS.breakMach[i] = bmBRKQuad;
                break;
            case scHardReturn:
                gbrS.breakMach[i] = bmBRKHardReturn;
                break;
            case scField:
                gbrS.breakMach[i] = bmBRKField;
                break;
            default:
                gbrS.breakMach[i] = bmBRKChar;
                break;
        }
    }
}

/**************************************************************************/

static Bool BRKStillMoreChars( CharRecordP  chRec,
                               long         count )
{
    for ( ; count--; chRec++ ) {
        if ( CTIsVisible( chRec->character ) )
            return true;
    }
    return false;
}
```

```
}

/*************************************************************************/

static void BRKAdjustWordSpace( CharRecordP prevChar,
                                GlyphSize   adjustment,
                                long        numSpaces,
                                long        endSpaces )
{
    /* when we come in here prevchar points to the first word of the next line
     * we need to ignore it if it is a wordspace
     */
    if (  prevChar->character  == scWordSpace )
        prevChar--;

    for ( ; endSpaces && prevChar > gbrS.gStartRec; prevChar-- ) {
        if (  prevChar->character  == scWordSpace )
            endSpaces--;
    }
    scAssert( endSpaces == 0 );
    for ( ; numSpaces && prevChar >= gbrS.gStartRec; prevChar-- ) {
        if (  prevChar->character  == scWordSpace ) {
            prevChar->escapement = adjustment;
            numSpaces--;
        }
    }
    scAssert( numSpaces == 0 );
}

/*************************************************************************/

static void BRKRepairFinalSpace(  )
{
    scAssert( gbrS.cB.theChRec->character == 0 );
    BRKRepairLastSpace( gbrS.cB.theChRec, gbrS.cB.trailingSpaces );
}

/*************************************************************************/

BreakStruct::BreakStruct()
{
}

/*************************************************************************/

BreakStruct::~BreakStruct()
{
}

/*************************************************************************/

void BreakStruct::Init()
{
    pspec_.clear();
    cB.Init();
    for ( int i = 0; i < MAXBREAKVALS; i++ )
        gbrS.candBreak[i].Init();
}

/*************************************************************************/
/* free the memory associated with the breaking machine */

void BRKFreeMach(  )
{
    delete [] gbrS.breakMach,           gbrS.breakMach      = 0;
    delete [] gbrS.fMaxLineVals,        gbrS.fMaxLineVals   = 0;
    delete [] gbrS.candBreak,           gbrS.candBreak      = 0;
}

/*************************************************************************/
/* init the breaking machine */

void BRKInitMach(  )
```

```
    if (  gbrS.cB.lineVal + 1 < MAXLEADVALS ) {
        gbrS.cB.specChanged++;
        mlvIndex = gbrS.cB.lineVal;

        gbrS.fMaxLineVals[mlvIndex].fSpecRec     = specRecEntry;
        gbrS.fMaxLineVals[mlvIndex].fMaxLead.Set( scCachedStyle::GetCurrentCache().GetComputedLead()
 );

        gbrS.fMaxLineVals[mlvIndex].fMaxInkExtents = scCachedStyle::GetCurrentCache().GetInkExtents(
);

        gbrS.fMaxLineVals[mlvIndex++].fOblique   = scCachedStyle::GetCurrentCache().GetHorzOblique();

        gbrS.cB.lineVal = mlvIndex;

        *( gbrS.fMaxLineVals + gbrS.cB.lineVal ) = gbrS.fZeroMaxLineVals;
        gbrS.cB.specRec = specRecEntry;
    }

    return theSpec;
}

/***************************************************************************/
/* find the last non-space character on the line, given that what is passed in
 * is the last character on the line
 */

static CharRecordP BRKLastCharOnLine( CharRecordP tmpChRec )
{
    for ( ; CTIsSpace( tmpChRec->character ); tmpChRec-- )
        ;
    return tmpChRec;
}

/***************************************************************************/

static void BRKRepairLastSpace( CharRecordP tmpChRec,
                                long        numberToNull )
{
    switch ( (tmpChRec-1)->character ) {
        case scQuadCenter:
            gbrS.effectiveRag = eRagCentered;
            tmpChRec -= 2;
            break;
        case scQuadLeft:
            gbrS.effectiveRag = eRagRight;
            tmpChRec -= 2;
            break;
        case scQuadRight:
            gbrS.effectiveRag = eRagLeft;
            tmpChRec -= 2;
            break;
        case scQuadJustify:
            gbrS.effectiveRag = eRagJustified;
            tmpChRec -= 2;
            break;
        case scHardReturn:
        case scVertTab:
            tmpChRec -= 2;
            break;
        default:
            tmpChRec--;
            break;
    }

    gbrS.totalTrailingSpace = 0;
    for ( ; numberToNull && tmpChRec->character == scWordSpace; tmpChRec--, numberToNull-- ) {
        if ( gHiliteSpaces )
            gbrS.totalTrailingSpace += tmpChRec->escapement;
    }

    scAssert( !numberToNull );
```

```
        gbrS.theBreakColH   = ggcS.theActiveColH;
        gbrS.dcLastBaseline = LONG_MIN;
    }
    else
        gbrS.dcSet = false;

    dcLeftOffset = 0;
    if ( gbrS.dcInfo.dcLineOrgChange && gbrS.dcLastBaseline != y ) {
        if ( y > gbrS.dcInfo.dcVMax || ggcS.theActiveColH != gbrS.theBreakColH )
            gbrS.dcInfo.dcLineOrgChange = 0;
        else            /* need to compute left indent for drop caps */
            dcLeftOffset = gbrS.dcInfo.dcLineOrgChange - x;
    }

        // INDENTION CONTROL
    if ( gbrS.effectiveRag & (int)eRagRight && (long)lineCount < scCachedStyle::GetParaStyle().GetLi
nesToIndent() )
        gbrS.brkLeftMargin = scCachedStyle::GetParaStyle().GetIndentAmount() + dcLeftOffset;
    else
        gbrS.brkLeftMargin = dcLeftOffset;


    if ( gbrS.dcLastBaseline != y && x <= gbrS.charIndent )
        gbrS.brkLeftMargin += ( gbrS.charIndent - x );


        // HANGING PUNCTUATION CONTROL
        // this computes the actual overhang
    if ( ( gbrS.effectiveRag & (int)eFlushLeft )
            && ( gbrS.effectiveRag & (int)eHangPuncLeft )
            && CTIsPunc( chRec->character ) )
        gbrS.brkLeftMargin -= scCachedStyle::GetParaStyle().GetLeftHangValue( chRec->character );


    gbrS.theBreakColH   = ggcS.theActiveColH;
    gbrS.dcLastBaseline = y;

        // compute the desired measure
    dMeasure = measure - gbrS.brkLeftMargin - gbrS.brkRightMargin;

        // compute the hyphenation zone
    if ( scCachedStyle::GetParaStyle().GetRagZone() > dMeasure )
        gbrS.hyphenationZone = dMeasure / 2;
    else
        gbrS.hyphenationZone = scCachedStyle::GetParaStyle().GetRagZone();


#if 0
    /* A LITTLE BULLET PROOFING
     * NOTE: it will be so out of whack the user will spot it fast
     */
    if ( dMeasure < 0 ) {
            SysBeep(10);
            return 0;
            return one_point;
    }
#endif

    return dMeasure;
}

/**************************************************************************/

static TypeSpec BRKUpdateSpec( scSpecRecord *specRecEntry )
{
    TypeSpec    theSpec     = specRecEntry->spec();
    size_t      mlvIndex;

    scCachedStyle::GetCachedStyle( theSpec );

        /* this is to take care of the rag setting on a line */
    if (  gbrS.cB.startCount  == (  gbrS.cB.streamCount  - 1 ) )
         gbrS.effectiveRag  =  scCachedStyle::GetParaStyle().GetRag() ;
    }
}

/**********************************************************************/
/* we have hit end of stream, check to see if we have exceeded measure,
 * if not longjmp out, otherwise back up to a reasonable break point
 * and get out
 */


static eBreakEvent bmBRKEndStream(  )
{
    MicroPoint adjustableSpace
                    =  gbrS.desiredMeasure - gbrS.cB.curBox;

    if ( gbrS.cB.maxGlue > adjustableSpace ) {
        BRKSetCandBreak(  eEndStreamBreak );
            if ( gbrS.cB.minGlue > adjustableSpace ) {
                BRKLineDecision( 0 );
                return BRKExitLoop(  );
            }
        }

    if ( gbrS.cB.optGlue > adjustableSpace ) {
        BRKSetCandBreak(  eEndStreamBreak );
        BRKLineDecision( 0 );
        return BRKExitLoop(  );
    }
    BRKSetCandBreak(  eEndStreamBreak );

    if ( BRKLineDecision( 0 ) == scEndStream )
        return end_of_stream_reached;

    return measure_exceeded;
}

/**********************************************************************/
/* this sets up the linebreaker by initing the spec, performing indents,
 * rag zone control, etc. returns the desired measure of the line
 */


static MicroPoint BRKRagControl( CharRecordP    chRec,
                                 MicroPoint     x,
                                 MicroPoint     y,
                                 MicroPoint     measure,
                                 TypeSpec       spec,
                                 ushort         lineCount,
                                 short          linesHyphenated )
{
    MicroPoint  dcLeftOffset;
    MicroPoint  dMeasure;

    scCachedStyle::GetCachedStyle(  spec );
    gbrS.effectiveRag    = scCachedStyle::GetParaStyle().GetRag();
    gbrS.brkRightMargin = scCachedStyle::GetParaStyle().GetRightBlockIndent();
    gbrS.theLineOrg     = x;

        // CONSECUTIVE HYPHENATED LINE CONTROL
    if ( scCachedStyle::GetParaStyle().GetHyphenate() && linesHyphenated < scCachedStyle::GetParaSty
le().GetMaxConsHyphs() )
        gbrS.allowHyphens = true;
    else
        gbrS.allowHyphens = false;

    gbrS.pspec_ = scCachedStyle::GetParaStyle().GetSpec();

        // DROP CAP CONTROL
    if ( lineCount == 0 ) {
        gbrS.cB.spec        = ::BRKUpdateSpec( gbrS.theSpecRec );
        gbrS.theSpecRec++;
        ::BRKDropCapControl( x, y );
    gbrS.fNoStartline = false;
    gbrS.fLastHangable = 0;

    gbrS.cB.streamCount++;
    gbrS.cB.fillSpCount++;
    gbrS.cB.theChRec->escapement = 0;
    gbrS.cB.theChRec++;

    return in_line;
}

/**************************************************************************/

static eBreakEvent bmBRKHyphen(  )
{
    MicroPoint adjustableSpace;

    if ( gbrS.firstBox ) {
        adjustableSpace =  gbrS.desiredMeasure - gbrS.cB.curBox;

        BRKSetCandBreak(  eCharBreak );
        if ( gbrS.cB.minGlue > adjustableSpace ) {
            BRKLineDecision( 0 );
            return BRKExitLoop(  );
        }
        BRKSetFirstBox();
//      gbrS.firstGlue          = true;
//      gbrS.firstBox           = false;
//      gbrS.cB.minGlue         += gbrS.tmpMinGlue;
//      gbrS.cB.optGlue         += gbrS.tmpOptGlue;
//      gbrS.cB.maxGlue         += gbrS.tmpMaxGlue;
//      gbrS.tmpMinGlue         = gbrS.tmpOptGlue = gbrS.tmpMaxGlue = 0;
//      gbrS.cB.trailingSpaces  = 0;
    }

    gbrS.fNoStartline = false;
    gbrS.fLastHangable = 0;

    gbrS.cB.curBox += gbrS.cB.theChRec->escapement;

    gbrS.cB.chCount++;
    gbrS.cB.streamCount++;
    gbrS.cB.theChRec++;

    BRKSetCandBreak(  eHardHyphBreak );

    return in_line;
}

/**************************************************************************/
/* start the stream */

static void BRKDropCapControl( MicroPoint   lineOrg,
                               MicroPoint   baseline )
{
    int visible     = CTIsDropCapable( gbrS.cB.theChRec->character )
                        && scCachedStyle::GetParaStyle().GetFlowdir().IsHorizontal();
    int flushleft   = gbrS.effectiveRag & (int)eRagRight;
    if (  visible && flushleft && ::DCCompute( gbrS.dcInfo,
                                               gbrS.pspec_,
                                               gbrS.cB.spec,
                                               lineOrg,
                                               baseline,
                                               gbrS.cB.theChRec->character ) ) {
            gbrS.cB.theChRec->flags.SetDropCap();
            gbrS.dcSet = true;
            gbrS.cB.streamCount++;
            gbrS.cB.theChRec++;
    }
    else {
        gbrS.dcSet = false;
        SCmemset( &gbrS.dcInfo, 0, sizeof( DropCapInfo ) );
        gbrS.cB.theChRec->flags.ClrDropCap();
    gbrS.firstGlue      = true;
    gbrS.firstBox       = true;     /* sil to true on 6/12/92 */
    gbrS.fNoStartline   = false;
    gbrS.fLastHangable  = 0;

    gbrS.cB.minGlue = gbrS.cB.optGlue = gbrS.cB.maxGlue = 0;
    gbrS.tmpMinGlue = gbrS.tmpOptGlue = gbrS.tmpMaxGlue = 0;
    gbrS.cB.trailingSpaces  = 0;
    gbrS.cB.wsSpaceCount    = 0;
    gbrS.cB.spaceCount      = 0;

        // define noLeftAlignTabbedLines and this
        // will allow tabbed lines to be none left aligned,
        // the manager of the spec system had better
        // guarantee that the values are reasonable
#ifndef noLeftAlignTabbedLines
    gbrS.cB.fillSpCount++;
#endif

    return in_line;
}

/*************************************************************************/

static eBreakEvent bmBRKRule( void )
{
    if ( gbrS.cB.curBox ) {
        BRKSetCandBreak( eCharBreak );
        BRKLineDecision( 0 );
        return BRKExitLoop(  );
    }

    CharRecordP chRec = gbrS.cB.theChRec;

    chRec->escapement = gbrS.desiredMeasure;

    gbrS.cB.curBox  +=  gbrS.cB.theChRec->escapement;
    gbrS.cB.streamCount++;
    gbrS.cB.theChRec++;

    if ( gbrS.cB.theChRec->character ) {
        BRKSetCandBreak(  eSpaceBreak );
        BRKLineDecision( 0 );
        return BRKExitLoop(  );
    }
    return in_line;
}

/*************************************************************************/

static eBreakEvent bmBRKFillSpace(  )
{
    MicroPoint adjustableSpace;

    if ( gbrS.firstBox ) {
        adjustableSpace =  gbrS.desiredMeasure - gbrS.cB.curBox;

        BRKSetCandBreak(  eCharBreak );
        if ( gbrS.cB.minGlue > adjustableSpace ) {
            BRKLineDecision( 0 );
            return BRKExitLoop(  );
        }
        BRKSetFirstBox();
//      gbrS.firstGlue          = true;
//      gbrS.firstBox           = false;
//      gbrS.cB.minGlue         += gbrS.tmpMinGlue;
//      gbrS.cB.optGlue         += gbrS.tmpOptGlue;
//      gbrS.cB.maxGlue         += gbrS.tmpMaxGlue;
//      gbrS.tmpMinGlue         = gbrS.tmpOptGlue = gbrS.tmpMaxGlue = 0;
//      gbrS.cB.trailingSpaces  = 0;
    }
```

```
                    indent += chRec->escapement;
                break;
            case scFixRelSpace:
                indent += SCRLUCompMP( scCachedStyle::GetCurrentCache().GetGlyphWidth(), (RLU)chRec-
>escapement );
                break;
        }
    }
    gbrS.foundCharIndent = false;
}

/**************************************************************************/

static eBreakEvent bmBRKTab(  )
{
    scTabInfo   tabInfo;
    MicroPoint  currentPosition,
                nextTokenWidth  = 0,
                alignTokenWidth = 0;
    CharRecordP tabChRec        = gbrS.cB.theChRec;

    BRKSetCandBreak(  eCharBreak );

    tabChRec->escapement = 0;
    currentPosition = gbrS.cB.curBox +
                      gbrS.cB.optGlue + gbrS.tmpOptGlue +
                      gbrS.brkLeftMargin + gbrS.theLineOrg;

    TSTabInfo( gbrS.pspec_,
               gbrS.cB.spec,
               tabInfo,
               currentPosition,
               0,
               gbrS.theLineCount );

    switch ( tabInfo.tabAlign ) {
        default:
        case eTBLeftAlign:
            break;
        case eTBRightAlign:
            alignTokenWidth = nextTokenWidth = BRKNextTokenWidth( gbrS.cB.theChRec+1,'\0');
            break;
        case eTBDecimalAlign:
            alignTokenWidth = BRKNextTokenWidth( gbrS.cB.theChRec + 1, scCachedStyle::GetCurrentCach
e().GetDecimalChar() );

            nextTokenWidth = BRKNextTokenWidth( gbrS.cB.theChRec+1,'\0');
            break;
        case eTBCenterAlign:
            nextTokenWidth = BRKNextTokenWidth( gbrS.cB.theChRec + 1, '\0' );
            alignTokenWidth = scRoundMP( (REAL)nextTokenWidth / 2 );
            break;
    }

    tabChRec->escapement = (GlyphSize)(tabInfo.xPosition - currentPosition - alignTokenWidth);

    if ( tabChRec->escapement < 0 ) {
        alignTokenWidth     += tabChRec->escapement;        /* wam added 7/22 */
        tabChRec->escapement = 0;
    }

    if ( gbrS.desiredMeasure < currentPosition + tabChRec->escapement + nextTokenWidth ) {
        if ( gbrS.cB.curBox + gbrS.cB.optGlue + gbrS.tmpOptGlue > 0 ) {
            BRKLineDecision( 0 );
            return BRKExitLoop(  );
        }
    }

    gbrS.cB.curBox = tabInfo.xPosition - (  gbrS.brkLeftMargin +  gbrS.theLineOrg  ) - alignTokenWid
th;

    gbrS.cB.theChRec++;
    gbrS.cB.streamCount++;
```

```
                case scHardReturn:
                case scQuadCenter:
                case scQuadLeft:
                case scQuadRight:
                case scQuadJustify:
                    return true;
            }
        }
    }

/**********************************************************************/

    static MicroPoint   BRKNextTokenWidth( CharRecordP  chRec,
                                           UCS2         breakCh )
    {
        MicroPoint tokenWidth = 0;
        MicroPoint charWidth;
        UCS2    theCh;
        long            tStreamCount    = gbrS.cB.streamCount;
        scSpecRecord *      curSpecRec      = gbrS.theSpecRec;

        for ( theCh = chRec->character;
                !TabBreakChar(theCh,breakCh);
                chRec++,theCh = chRec->character ) {

            if ( (long)tStreamCount >= gbrS.theSpecRec->offset() ) {
                gbrS.cB.spec
                    = BRKUpdateSpec(  gbrS.theSpecRec );
                gbrS.theSpecRec++;
            }
            switch ( theCh ) {
                default:
                    charWidth = chRec->escapement;
                    break;
                case scWordSpace:
                    charWidth = scCachedStyle::GetCurrentCache().GetOptWord();
                    break;
                case scFixRelSpace:
                    charWidth = SCRLUCompGS( scCachedStyle::GetCurrentCache().GetSetSize(),(RLU)chRe
c->escapement );
                    break;
            }

            tokenWidth += charWidth;
        }

        if ( curSpecRec != gbrS.theSpecRec ) {
            gbrS.theSpecRec = curSpecRec;
            gbrS.cB.spec
                = BRKUpdateSpec(  gbrS.theSpecRec );
        }

        return tokenWidth;
    }


/**********************************************************************/

static void BRKSetCharIndent(

    CharRecordP chRec,  /* the character array */
    long    startCount, /* count into char array that starts line */
    long    count,      /* count into char array of end of line */
    MicroPoint  letterSpace )
{
    MicroPoint indent;

    for ( indent = 0, chRec += startCount; count--; chRec++ ) {
        switch ( chRec->character ) {
            default:
                if ( LETTERSPACE( chRec ) )
                    indent += (chRec->escapement + letterSpace);
                else
    gbrS.fNoStartline = false;
    gbrS.fLastHangable = 0;

    gbrS.cB.curBox += SCRLUCompMP( scCachedStyle::GetCurrentCache().GetGlyphWidth(), (RLU) gbrS.cB.t
heChRec->escapement );
    gbrS.minRelPosition = MIN( gbrS.cB.curBox, gbrS.minRelPosition );
    gbrS.cB.theChRec++;
    gbrS.cB.streamCount++;

    return in_line;
}

/***************************************************************************/

static eBreakEvent bmBRKHardReturn(  )
{
    gbrS.cB.theChRec->escapement = 0;
    gbrS.cB.theChRec++;
    gbrS.cB.streamCount++;

    BRKSetCandBreak(  eSpaceBreak );
    BRKLineDecision( 0 );
    return BRKExitLoop(  );
}

/***************************************************************************/

static eBreakEvent bmBRKQuad(  )
{
    gbrS.cB.theChRec->escapement = 0;
    gbrS.cB.theChRec++;
    gbrS.cB.streamCount++;
    if ( gbrS.cB.theChRec->character == scEndStream )
        BRKSetCandBreak(  eEndStreamBreak );
    else
        BRKSetCandBreak( eSpaceBreak );

    BRKLineDecision( 0 );
    return BRKExitLoop(  );
}

/***************************************************************************/

static eBreakEvent bmBRKVertTab(  )
{
    gbrS.cB.theChRec->escapement = 0;
    gbrS.cB.theChRec++;
    gbrS.cB.streamCount++;
    BRKSetCandBreak( eColumnBreak );

    BRKLineDecision( 0 );
    return BRKExitLoop(  );
}

/***************************************************************************/

    static Bool TabBreakChar( UCS2 theCh,
                              UCS2 breakCh )
    {
        if ( breakCh == theCh )
            return true;
        else {
            switch (  theCh  ) {
                default:
                    return false;
                case scEndStream:

/*              case wordSpace:*/
                case scTabSpace:
                case scFillSpace:
                /* vetical breaks */
                case scVertTab:
                /* horizontal breaks */
            gbrS.letterSpaceAdj = glueSpace;

#ifdef LimitLetterSpace
        /* should we constrain this to min/max letterspace */
        gbrS.letterSpaceAdj = MIN( gbrS.letterSpaceAdj, scCachedStyle::GetCurrentCache().GetMaxLSP()
 );
        gbrS.letterSpaceAdj = MAX( gbrS.letterSpaceAdj, scCachedStyle::GetCurrentCache().GetMinLSP()
 );
#endif
    }
}

/*************************************************************************/

static eBreakEvent bmBRKFixSpace(  )
{
    MicroPoint adjustableSpace;

    if ( gbrS.firstBox ) {
        adjustableSpace =  gbrS.desiredMeasure - gbrS.cB.curBox;

            /* at the start of every word set a potential break point */
        BRKSetCandBreak(  eCharBreak );
        if ( BRKExceedVals(  adjustableSpace ) ) {
            BRKLineDecision( 0 );
            return BRKExitLoop(  );
        }
        BRKSetFirstBox();

//      gbrS.firstGlue            = true;
//      gbrS.firstBox             = false;
//      gbrS.cB.minGlue           += gbrS.tmpMinGlue;
//      gbrS.cB.optGlue           += gbrS.tmpOptGlue;
//      gbrS.cB.maxGlue           += gbrS.tmpMaxGlue;
//      gbrS.tmpMinGlue           = gbrS.tmpOptGlue = gbrS.tmpMaxGlue = 0;
//      gbrS.cB.trailingSpaces    = 0;
    }

    gbrS.fNoStartline = false;
    gbrS.fLastHangable = 0;

    gbrS.cB.curBox += gbrS.cB.theChRec->escapement;
    gbrS.cB.theChRec++;
    gbrS.cB.streamCount++;

    return in_line;
}

/*************************************************************************/

static eBreakEvent bmBRKRelSpace(  )
{
    MicroPoint adjustableSpace;

    if ( gbrS.firstBox ) {
        adjustableSpace =  gbrS.desiredMeasure - gbrS.cB.curBox;

        /* at the start of every word set a potential break point */
        BRKSetCandBreak(  eCharBreak );
        if ( BRKExceedVals(  adjustableSpace ) ) {
            BRKLineDecision( 0 );
            return BRKExitLoop(  );
        }
        BRKSetFirstBox();
//      gbrS.firstGlue            = true;
//      gbrS.firstBox             = false;
//      gbrS.cB.minGlue           += gbrS.tmpMinGlue;
//      gbrS.cB.optGlue           += gbrS.tmpOptGlue;
//      gbrS.cB.maxGlue           += gbrS.tmpMaxGlue;
//      gbrS.tmpMinGlue           = gbrS.tmpOptGlue = gbrS.tmpMaxGlue = 0;
//      gbrS.cB.trailingSpaces    = 0;
    }
    if ( !gbrS.lineHyphenated )
        glueSpace += scCachedStyle::GetCurrentCache().GetOptLSP();

    if ( gbrS.cB.spaceCount ) {

        gbrS.justSpace = scRoundGS( (REAL)glueSpace / gbrS.cB.spaceCount );

        if ( gbrS.justSpace < 0 ) {
            lspSpaces       = COMP_LETTERSPACES( gbrS.cB.chCount, 0, gbrS.cB.theChRec);
            if ( lspSpaces )
                gbrS.letterSpaceAdj = scRoundGS( (REAL)glueSpace / lspSpaces );
            else
                gbrS.letterSpaceAdj = glueSpace;
            gbrS.justSpace      = 0;
        }
        else if ( gbrS.justSpace > scCachedStyle::GetCurrentCache().GetMaxWord() ) {
            lspSpaces       = COMP_LETTERSPACES( gbrS.cB.chCount, gbrS.cB.spaceCount, gbrS.cB.theCh
Rec);
            adjustableSpace = MPtoGS( glueSpace - gbrS.cB.maxGlue );
            if ( lspSpaces )
                gbrS.letterSpaceAdj = scRoundGS( (REAL)adjustableSpace / lspSpaces );
            else
                gbrS.letterSpaceAdj = adjustableSpace;
            if ( gbrS.letterSpaceAdj < scCachedStyle::GetCurrentCache().GetMinLSP() ) {
                gbrS.letterSpaceAdj = scCachedStyle::GetCurrentCache().GetMinLSP();
                adjustableSpace = MPtoGS( glueSpace - scRoundMP( (REAL)gbrS.letterSpaceAdj * lspSpac
es ) );
                gbrS.justSpace = scRoundGS( (REAL)adjustableSpace / gbrS.cB.spaceCount );
            }
            else if ( gbrS.letterSpaceAdj > scCachedStyle::GetCurrentCache().GetMaxLSP() ) {
                gbrS.letterSpaceAdj = scCachedStyle::GetCurrentCache().GetMaxLSP();
                adjustableSpace = MPtoGS( glueSpace - scRoundMP( (REAL)gbrS.letterSpaceAdj * lspSpac
es ) );
                gbrS.justSpace = scRoundGS( (REAL)adjustableSpace / gbrS.cB.spaceCount );
            }
            else
                gbrS.justSpace = scCachedStyle::GetCurrentCache().GetMaxWord();
        }
        else if ( gbrS.justSpace < scCachedStyle::GetCurrentCache().GetMinWord() ) {
            lspSpaces       = COMP_LETTERSPACES( gbrS.cB.chCount, gbrS.cB.spaceCount, gbrS.cB.theChR
ec);
            adjustableSpace = MPtoGS( glueSpace - gbrS.cB.minGlue );

            if ( lspSpaces )
                gbrS.letterSpaceAdj = scRoundGS( (REAL)adjustableSpace / lspSpaces );
            else
                gbrS.letterSpaceAdj = adjustableSpace;

            if ( gbrS.letterSpaceAdj < scCachedStyle::GetCurrentCache().GetMinLSP() ) {
                gbrS.letterSpaceAdj = scCachedStyle::GetCurrentCache().GetMinLSP();
                adjustableSpace = MPtoGS( glueSpace - scRoundMP( (REAL)gbrS.letterSpaceAdj * lspSpac
es ) );
                gbrS.justSpace = scRoundGS( (REAL)adjustableSpace / gbrS.cB.spaceCount );
            }
            else if ( gbrS.letterSpaceAdj > scCachedStyle::GetCurrentCache().GetMaxLSP() ) {
                gbrS.letterSpaceAdj = scCachedStyle::GetCurrentCache().GetMaxLSP();
                adjustableSpace = MPtoGS( glueSpace - scRoundMP( (REAL)gbrS.letterSpaceAdj / lspSpac
es ) );
                gbrS.justSpace  = scRoundGS( (REAL)adjustableSpace / gbrS.cB.spaceCount );
            }
            else
                gbrS.justSpace = scCachedStyle::GetCurrentCache().GetMinWord();
        }
        BRKAdjustWordSpace( gbrS.cB.theChRec,
                            gbrS.justSpace, gbrS.cB.spaceCount,
                            gbrS.cB.trailingSpaces );
    }
    else {
        lspSpaces       = COMP_LETTERSPACES( gbrS.cB.chCount, 0, gbrS.cB.theChRec);
        if ( lspSpaces )
            gbrS.letterSpaceAdj = scRoundGS( (REAL)glueSpace / lspSpaces );
        else
        case eRagLeft:
            if ( gbrS.effectiveRag & (int)eHangPuncRight )
                 gbrS.desiredMeasure += BRKHangPuncRightAdjust(  );
            translation =  gbrS.desiredMeasure - actualMeasure + gbrS.brkLeftMargin;

            if ( !gbrS.lineHyphenated ) {
                /* this accounts for any track kerning - no need to worry about with
                 * hyphenation because, we want trackkerning between the hyphen
                 * and the last character - and the hyphen escapement does not
                 * include any track kerning
                 */
                translation += scCachedStyle::GetCurrentCache().GetOptLSP();
            }
            break;
        case eRagCentered:
            if ( gbrS.lineHyphenated )
                translation = scRoundMP( (REAL)( gbrS.desiredMeasure - actualMeasure) / 2 ) + gbrS.b
rkLeftMargin;
            else
                translation = scRoundMP( (REAL)( gbrS.desiredMeasure - actualMeasure + scCachedStyle
::GetCurrentCache().GetOptLSP()) / 2 ) + gbrS.brkLeftMargin;
            break;
    }

    if ( fd.IsHorizontal() )
        lineOrigin.Translate( translation, 0 );
    else
        lineOrigin.Translate( 0, translation );

    if ( gHiliteSpaces )
        actualMeasure += gbrS.totalTrailingSpace;

    measure = actualMeasure;
}

/*******************************************************************/
/* handle adjustment for hanging punctuation on the right */

static MicroPoint BRKHangPuncRightAdjust(  )
{
    CharRecordP lastCharOnLine;

    if ( gbrS.lineHyphenated )
        return scCachedStyle::GetCurrentCache().GetRightHangValue( scCachedStyle::GetCurrentCache().
GetHyphChar() );

    lastCharOnLine = BRKLastCharOnLine( gbrS.cB.theChRec - 1 );
    if ( CTIsPunc(  lastCharOnLine->character  ) )
        return scCachedStyle::GetCurrentCache().GetRightHangValue( lastCharOnLine->character );
    else
        return 0L;
}

/*******************************************************************/

#define COMP_LETTERSPACES( charCount, spaceCount, chRec ) \
    ((long)( charCount-spaceCount-1 + ((chRec-1)->flags.IsHyphPresent() ? 1:0) ))

static void BRKJustifyLine(  )
{
    GlyphSize   glueSpace,
                adjustableSpace;
    long        lspSpaces;

    if ( gbrS.effectiveRag & (int)eHangPuncRight )
         gbrS.desiredMeasure += BRKHangPuncRightAdjust(  );

    glueSpace =  (GlyphSize)(gbrS.desiredMeasure - gbrS.cB.curBox);

    if ( glueSpace < 0 ) {
        gbrS.desiredMeasure += gbrS.cB.fHangable;
        glueSpace =  (GlyphSize)(gbrS.desiredMeasure - gbrS.cB.curBox);
    }
```

```
/***********************************************************************/

static CandBreak *BRKLineDecisionJust(  )
{
    CandBreak   *theBreak,
                *choice;
    long    bCount          = gbrS.cB.breakCount;
    MicroPoint  lineSpace,
                diff,
                bestDiff        = LONG_MAX;

    choice      = theBreak  = gbrS.candBreak + bCount - 1;

    for ( ; bCount-- > 0; theBreak-- ) {
        lineSpace   = theBreak->curBox + theBreak->minGlue;
        /* this reflects space changed by hyphenation spelling changes */

        diff        = lineSpace - gbrS.desiredMeasure;
        if ( diff < theBreak->fHangable && ABS( diff ) < bestDiff ) {
            choice      = theBreak;
            bestDiff    = ABS( diff );
        }
        if ( diff > bestDiff )
            break;
    }

    return choice;
}
/***********************************************************************/
/* if we break on a hyphenation point and the character is kerned with
 * the next character we have an incorrect line length, because of the
 * kern built into the characters escapement, here we get that correction
 */

static MicroPoint BRKKernCorrection( CharRecordP aChRec )
{
    return ( scCachedStyle::GetCurrentCache().GetEscapement( aChRec->character ) - aChRec->escapemen
t );
}

/***********************************************************************/
/* for non-justified lines this places the line */

static void BRKPlaceLine( scMuPoint&           lineOrigin,
                          MicroPoint&       measure,
                          const scFlowDir&  fd )
{
    MicroPoint  actualMeasure,
                translation;

    actualMeasure = gbrS.cB.curBox + gbrS.cB.optGlue;
    if ( gbrS.cB.fillSpCount ) {
        MicroPoint  fill;
        long        count;
        CharRecordP tmpChRec;

        fill = gbrS.desiredMeasure - actualMeasure;
        fill = scRoundMP( (REAL)fill / gbrS.cB.fillSpCount );

        actualMeasure =  gbrS.desiredMeasure;
        for ( tmpChRec = gbrS.cB.theChRec, count = gbrS.cB.fillSpCount;
                    tmpChRec >= gbrS.gStartRec && count; tmpChRec-- ) {
            if ( tmpChRec->character == scFillSpace )
                tmpChRec->escapement = (GlyphSize)fill;
        }
    }

    switch ( gbrS.effectiveRag & eRagFlag ) {
        default:
        case eRagRight:
            translation = gbrS.brkLeftMargin;
            break;
        return choice;

    theBreak    = startWord = endWord;
    bCount      = endCount;             /* restore its last value */
    for ( ; bCount-- > 0; theBreak-- ) {
        if ( NotHyphBreak( theBreak->breakVal )
                && theBreak->curBox + theBreak->optGlue <  gbrS.desiredMeasure  + theBreak->fHangabl
e ) {
            startWord   = theBreak;
            break;
        }
    }

    BRKAddHyphens(  startWord, endWord );
    return BRKLineDecisionJust(  );
}

/***************************************************************************/

static CandBreak *BRKLineDecisionRag(  )
{
    CandBreak   *theBreak,
                *choice         = NULL;
    long    bCount          = gbrS.cB.breakCount;
    MicroPoint  lineSpace;

    /* there is a very strange bug here in that the char plus hyphen may be
     * chosen instead of the entire word because the hyphen may be wider than
     * the trailing letter(s)
     *
     * Note: We only get here if hyphenationZone is off or we failed to find
     * a good non-hyphen break.
     */

    if ( scCachedStyle::GetCurrentCache().GetDiffRagZone() ) {
        theBreak    = gbrS.candBreak + gbrS.cB.breakCount - 1;

        for ( ; bCount-- > 0; theBreak-- ) {

            lineSpace   = theBreak->curBox + theBreak->optGlue;
            /* this reflects space changed by hyphenation spelling changes */

            if (  gbrS.desiredMeasure >= lineSpace && ABS( gbrS.lastLineLen - lineSpace ) > scCached
Style::GetCurrentCache().GetDiffRagZone() ) {
                choice      = theBreak;
                break;
            }
        }
    }

    /* resort to worst case if necessary */
    /* if !GetDiffRagZone() && !hyphenationZone, we will fall through to here */

    if ( choice == NULL ) {
        bCount      = gbrS.cB.breakCount;
        choice      = theBreak      = gbrS.candBreak + gbrS.cB.breakCount - 1;

        for ( ; bCount-- > 0; theBreak-- ) {

            lineSpace       = theBreak->curBox + theBreak->optGlue;
            /* this reflects space changed by hyphenation spelling changes */

            if (  gbrS.desiredMeasure + theBreak->fHangable >= lineSpace ) {
                choice      = theBreak;
                break;
            }
        }
    }

    return choice;
}
```

```
    if ( choice )
        return choice;

    /* First, get closest word break greater than desired measure */

    bCount      = gbrS.cB.breakCount;
    endWord     = theBreak  = gbrS.candBreak + gbrS.cB.breakCount - 1;
    for ( ; bCount-- > 0; theBreak-- ) {

        diff    = theBreak->curBox + theBreak->optGlue - gbrS.desiredMeasure;
        if ( diff <= 0 )
            break;

        if ( NotHyphBreak( theBreak->breakVal ) && diff < bestDiff ) {
            bestDiff    = diff;
            endWord     = theBreak;
        }
    }

    /* Next, get closest word break less than desired measure */

    startWord   = endWord;
    bCount++;               /* reset it to what it should be */
    for ( ; bCount-- > 0; theBreak-- ) {
        if ( NotHyphBreak( theBreak->breakVal ) ) {
            startWord   = theBreak;
            break;
        }
    }

    BRKAddHyphens(  startWord, endWord );
    return BRKLineDecisionRag(  );
}

/**************************************************************************/

static CandBreak *BRKHyphenateJust(  )
{
    CandBreak       *theBreak;
    CandBreak               *choice         = NULL,
                            *startWord      = NULL,
                            *endWord;
    long                bCount      = gbrS.cB.breakCount;
    long                        endCount;
    MicroPoint      adjustableSpace,
                            diff,
                            bestDiff        = LONG_MAX;

    endWord     = theBreak  = gbrS.candBreak + bCount - 1;
    for ( ; bCount-- > 0; theBreak-- ) {

        /* amount of space we have to play with */
        adjustableSpace =  gbrS.desiredMeasure + theBreak->fHangable - theBreak->curBox;

        diff = adjustableSpace - theBreak->optGlue;
        if ( diff <= 0 && NotHyphBreak( theBreak->breakVal ) ) {
            endWord     = theBreak;
            endCount    = bCount + 1;
        }

        diff = ABS( diff );

        if ( NotHyphBreak( theBreak->breakVal ) && diff < bestDiff ) {
            if ( adjustableSpace <= theBreak->maxGlue && adjustableSpace >= theBreak->minGlue ) {
                choice      = theBreak;
                bestDiff    = diff;
            }
        }
        if ( diff > bestDiff )
            break;
    }

    if ( choice )
```

```
            (gbrS.cB.theChRec-1)->flags.SetAutoHyphen( hyphens[i].rank );

        if ( ( gbrS.effectiveRag & eRagFlag ) == eRagJustified ) {
            if ( gbrS.cB.curBox + gbrS.cB.minGlue >  gbrS.desiredMeasure )
                break;
        }
        else {
            if ( gbrS.cB.curBox + gbrS.cB.optGlue >  gbrS.desiredMeasure )
                break;
        }
    }

#ifdef scForceBreakFirstWord
        /* if the first hyphen exceeded the measure, we may */
        /* want to force a break.                           */
    if ( i == 0 && startBreak == &gbrS.candBreak[ 0 ]
            && startBreak->streamCount == startBreak->startCount ) {
        BRKForceHyphens(  startBreak, endBreak );
        return;
    }
#endif

    savedEndBreak.breakCount    = gbrS.cB.breakCount;
    gbrS.candBreak[gbrS.cB.breakCount]  = savedEndBreak;

    if ( gbrS.cB.breakCount >= (MAXBREAKVALS-1) )
        ShuffleBreakCandidates();
    else
        gbrS.cB.breakCount++;
}

/************************************************************************/

static CandBreak *BRKHyphenateRag(  )
{
    CandBreak   *theBreak,
                *choice         = NULL;
    CandBreak   *startWord      = NULL,
                *endWord;
    long    bCount          = gbrS.cB.breakCount;
    MicroPoint  diff,
                bestDiff        = LONG_MAX,
                lineSpace;
    Bool        lineDiff        = true; /* irrelevant unless otherwise set */

    /* If there is a hyphenation zone, first try to find a non-hyphen break.
     * If that fails, find the start and end of the word straddling the
     * line break, call hyphenation routine, and call line decision to
     * find the best break.
     */

    if ( gbrS.hyphenationZone ) {
        theBreak    = gbrS.candBreak + gbrS.cB.breakCount - 1;

        for ( ; bCount-- > 0; theBreak-- ) {

            lineSpace   = theBreak->curBox + theBreak->optGlue;
            if ( scCachedStyle::GetCurrentCache().GetDiffRagZone() )
                lineDiff    = ABS(  gbrS.lastLineLen - lineSpace ) > scCachedStyle::GetCurrentCache(
).GetDiffRagZone();

            /* look for a non-hyphen break and a line with a sufficient diff zone */
            if ( NotHyphBreak( theBreak->breakVal )
                    &&  gbrS.desiredMeasure >= lineSpace
                    &&  gbrS.desiredMeasure <= lineSpace + gbrS.hyphenationZone
                    && lineDiff )
            {
                choice      = theBreak;
                break;
            }
        }
    }
```

```
}

#endif /* forceBreakFirstWord */

/*************************************************************************/

#define NotHyphBreak( c )        ( ((c)!=eHyphBreak) && ((c)!=eHardHyphBreak ) )
#define useBadHyphens            false

static void BRKAddHyphens( CandBreak    *startBreak,
                           CandBreak    *endBreak )
{
    CandBreak   savedEndBreak;
    CharRecordP startChRec;
    CharRecordP stopChRec;
    Hyphen      hyphens[64];
    short       i,
                offset,
                prevOffset      = 0,
                numHyphens;

    if ( startBreak == endBreak || endBreak->curBox + endBreak->optGlue <=  gbrS.desiredMeasure )
        return;

    if ( ! gbrS.allowHyphens || ! scCachedStyle::GetCurrentCache().GetHyphenate() ) {

#ifdef scForceBreakFirstWord
        if ( startBreak == &gbrS.candBreak[0] && startBreak->streamCount == startBreak->startCount )
            BRKForceHyphens( startBreak, endBreak );
#endif

        return;
    }

    savedEndBreak   = *endBreak;
    startChRec      = startBreak->theChRec;
    stopChRec       = endBreak->theChRec;

    numHyphens  = BRKPerformHyphenation(  startChRec, stopChRec, hyphens );
    if ( numHyphens == 0 ) {

#ifdef scForceBreakFirstWord
        if ( startBreak == &gbrS.candBreak[0] && startBreak->streamCount == startBreak->startCount )
            BRKForceHyphens(  startBreak, endBreak );
#endif

        return;
    }

    /* We are resetting the break machine to the start of the word. */
    /* The values from this point on in the machine will be over-   */
    /* written. The end of word break, since it is past the measure,*/
    /* will never be restored, and the final break will now be the  */
    /* last hyphen break.                                           */
    /* Use endBreak to return the last hyphen breakpoint we find.   */

    gbrS.tmpMinGlue = (startBreak+1)->minGlue - startBreak->minGlue;
    gbrS.tmpOptGlue = (startBreak+1)->optGlue - startBreak->optGlue;
    gbrS.tmpMaxGlue = (startBreak+1)->maxGlue - startBreak->maxGlue;
    gbrS.cB         = *startBreak;
    gbrS.firstBox   = true;         /* signal start of word */

    for ( i = 0; i < numHyphens; i++ ) {
        if ( ! useBadHyphens && hyphens[i].rank == eBadHyphRank )
            continue;

        offset  = hyphens[i].offset;
        for ( ; prevOffset < offset; prevOffset++ )
            BRKLoopBody(  );

        BRKSetCandBreak(  eHyphBreak );

        if ( !(gbrS.cB.theChRec-1)->flags.IsDiscHyphen() )
```

```
                            hyphens[numBreaks].rank = eGoodHyphRank;
                            break;
                        default:
                            hyphens[numBreaks].rank = eBadHyphRank;
                            break;
                    }
                    hyphens[numBreaks].offset   = charCount;
                    numBreaks++;
                }
                j++;
            }
        }
    }

    return numBreaks;
}

/*****************************************************************/
/* This is called only in the case where even the first word (or the    */
/* first legal portion of it before a hyphen) will not fit on the line. */
/* Therefore, we add a hyphen after every character to force a break.   */

#ifdef scForceBreakFirstWord

static void BRKForceHyphens( CandBreak  *startBreak,
                             CandBreak  *endBreak )
{
    CandBreak       savedEndBreak;

    savedEndBreak   = *endBreak;

    /* We are resetting the break machine to the start of the word. */
    /* The values from this point on in the machine will be over-   */
    /* written.                                                     */

    gbrS.tmpMinGlue = (startBreak+1)->minGlue - startBreak->minGlue;
    gbrS.tmpOptGlue = (startBreak+1)->optGlue - startBreak->optGlue;
    gbrS.tmpMaxGlue = (startBreak+1)->maxGlue - startBreak->maxGlue;
    gbrS.cB         = *startBreak;
    gbrS.firstBox   = true;         /* signal start of word */

    while ( gbrS.theSpecRec->offset() > gbrS.cB.streamCount ) // reset the spec to the start of the
word
        gbrS.theSpecRec--;

    while ( true ) {

        BRKLoopBody(  );

        if ( gbrS.cB.chCount == savedEndBreak.chCount )
            break;

        BRKSetCandBreak(  eHyphBreak );
        if ( scCachedStyle::GetCurrentCache().GetHyphLanguage() != Japanese )
            (gbrS.cB.theChRec - 1)->flags.SetAutoHyphen( eGoodHyphRank );

        if ( ( gbrS.effectiveRag & eRagFlag ) == eRagJustified ) {
            if ( gbrS.cB.curBox + gbrS.cB.minGlue >  gbrS.desiredMeasure )
                break;
        }
        else {
            if ( gbrS.cB.curBox + gbrS.cB.optGlue >  gbrS.desiredMeasure )
                break;
        }
    }

    savedEndBreak.breakCount    = gbrS.cB.breakCount;
    gbrS.candBreak[gbrS.cB.breakCount]  = savedEndBreak;

    if ( gbrS.cB.breakCount >= (MAXBREAKVALS-1) )
        ShuffleBreakCandidates();
    else
        gbrS.cB.breakCount++;
```

```
            numBreaks--;

    return numBreaks;
}

/*************************************************************************/

static short BRKPerformHyphenation( CharRecordP firstChRec,
                                    CharRecordP lastChRec,
                                    Hyphen*     hyphens )
{
    CharRecordP theChar;
    int     j;
    UCS2    ch;
    UCS2    hyphWord[64];
    short       hyphArray[64];
    short       hLen            = 0;
    Bool        hitLowerCase    = false;
    Bool        hitUpperCase    = false;
    short       numBreaks       = 0;
    short       charCount       = 1;

    SCmemset( hyphArray, 0, sizeof( short ) * 64 );
    SCmemset( hyphWord, 0, sizeof( UCS2 ) * 64 );

    for ( theChar = firstChRec; theChar <= lastChRec; theChar++ ) {
        ch  =  theChar->character;
        if ( ch < 256 && CTIsAlpha( ch ) )
            break;
    }

    for ( ; theChar <= lastChRec; theChar++, charCount++ ) {

        ch =  theChar->character;

        if ( ch == scBreakingHyphen ) {
            return 0;
        }
        else if ( theChar->flags.IsDiscHyphen() ) {
            return BRKPerformDiscHyphen(  firstChRec, lastChRec, hyphens );
        }
        else if ( ch < 256 && CTIsAlpha( ch ) ) {
            if ( hLen < 63 ) {
                if ( CTIsLowerCase( ch ) )
                    hitLowerCase    = true;
                else if ( CTIsUpperCase( ch ) )
                    hitUpperCase    = true;
                hyphWord[ hLen++ ]  = CTToLower( ch );
            }
        }
        else if ( ch != scFixRelSpace && ch != scFixAbsSpace ) {    /* hit delimiter */
            break;
        }
    }

    if ( hLen < scCachedStyle::GetCurrentCache().GetMaxWordHyph()
            || ( ! scCachedStyle::GetCurrentCache().GetAcronymHyphs() && ! hitLowerCase )
            || ( ! scCachedStyle::GetCurrentCache().GetCaseHyphs() && hitUpperCase ) )
        return 0;

    if ( HYFWord( hyphWord, hyphArray ) ) {
        theChar = firstChRec;
        for ( j = 0, charCount = 1; j < hLen; charCount++, theChar++ ) {
            ch = CTToLower(  theChar->character );
            if ( ch == hyphWord[j] ) {
                if ( hyphArray[j] && j >= ( scCachedStyle::GetCurrentCache().GetPreHyph() - 1 ) && j
 < ( hLen - scCachedStyle::GetCurrentCache().GetPostHyph() ) ) {
                    switch( hyphArray[j] & 0x3f ) {
                        case 1:
                        case 2:
                            hyphens[numBreaks].rank = eBestHyphRank;
                            break;
                        case 3:
```

```
static eBreakEvent BRKTheLoop(  )
{
    eBreakEvent bt;
    while ( ( bt = BRKLoopBody() ) == in_line )
        ;
    return bt;
}

/*************************************************************************/
/* handle breaking of characters that we cannot vector to with thw
 * 'breakMach' array of sub-routines
 */

#if 0
static void BRKSpecial( ushort theCharacter )
{
    switch ( theCharacter ) {
        case scFillSpace:
            bmBRKFillSpace(  );
            break;
        default:
            bmBRKChar();
            break;
    }
}
#endif

/*************************************************************************/

static short BRKPerformDiscHyphen( CharRecordP  theChar,
                                   CharRecordP  lastChRec,
                                   Hyphen       *hyphens )
{
    UCS2        ch;
    short       charCount       = 1,
                numBreaks       = 0;
    int         j;
    Bool        hitLowerCase    = false;
    Bool        hitUpperCase    = false;

    for ( ; theChar <= lastChRec; theChar++, charCount++ ) {

        ch =  theChar->character;

        if ( ch == scBreakingHyphen )
            return 0;

        if (  theChar->flags.IsDiscHyphen() ) {
            if ( charCount >= scCachedStyle::GetCurrentCache().GetPreHyph() ) {
                hyphens[numBreaks].offset   = charCount;
                hyphens[numBreaks].rank     = eDiscHyphRank;
                numBreaks++;
            }
        }

        if ( ch < 256 && CTIsAlpha( ch ) ) {
            if ( CTIsLowerCase( ch ) )
                hitLowerCase    = true;
            else if ( CTIsUpperCase( ch ) )
                hitUpperCase    = true;
        }
        else if ( CTIsSpace( ch ) )
            if ( ch != scFixRelSpace && ch != scFixAbsSpace )
                break;
    }

    if ( ( ! scCachedStyle::GetCurrentCache().GetAcronymHyphs() && ! hitLowerCase ) || ( ! scCachedS
tyle::GetCurrentCache().GetCaseHyphs() && hitUpperCase ) )
        return 0;

    charCount = (short)(charCount - scCachedStyle::GetCurrentCache().GetPostHyph());
    for ( j = numBreaks - 1; j >= 0 && numBreaks > 0; j-- )
        if ( hyphens[j].offset >= charCount )
        if ( BRKExceedVals(  adjustableSpace ) ) {
            BRKLineDecision( 0 );
            return BRKExitLoop(  );
        }

        BRKSetFirstBox(  );
    }

    if ( ustr.len )
        gbrS.cB.theChRec->escapement = UnivStringWidth( ustr, 0, spec );

    gbrS.cB.curBox  +=  gbrS.cB.theChRec ->escapement;

    gbrS.cB.chCount++;
    gbrS.cB.streamCount++;
    gbrS.cB.theChRec++;

    return in_line;
}

/***********************************************************************/

static inline eBreakEvent BRKLoopBody(  )
{
    register ushort theCharacter = gbrS.cB.theChRec->character;
    register eBreakEvent breaktype = in_line;

    /* increment the spec counter if necessary,
     * NOTE - this increments the spec too soon!!!
     * If the next character is the break char then
     * the spec will be inaccurate, there is a fix
     * for this in the line ending decision logic,
     * it is commented as such, the cleaner fix
     * would slow the code down too much
     * (sorry for any confusion WAM)
     */
    if ( gbrS.cB.streamCount >= gbrS.theSpecRec->offset() ) {
        if ( theCharacter != scEndStream ) {
            /* do not advance spec unless we have characters */
            gbrS.cB.spec = BRKUpdateSpec( gbrS.theSpecRec );
            gbrS.theSpecRec++;
        }
    }

    /* dispatch the character to the appropriate routine */

    gbrS.cB.theChRec->flags.ClrVarious();

    if ( ! ( theCharacter & 0xFFC0 ) ) {
        if ( theCharacter == scWordSpace )
            breaktype = bmBRKWordSpace();
        else if ( theCharacter & 0x0030 ) {
            switch ( theCharacter ) {
            case scBreakingHyphen:
                breaktype = bmBRKHyphen();
                break;
            default:
                breaktype = bmBRKChar();
                break;
            }
        }
        else
            breaktype = (*gbrS.breakMach[theCharacter])();
    }
    else
        breaktype = bmBRKChar();

    return breaktype;
}

/***********************************************************************/
```

```
{
    MicroPoint      adjustableSpace;

    if ( gbrS.cB.theChRec->character >= 256 ) {
        adjustableSpace = gbrS.desiredMeasure - gbrS.cB.curBox;
        BRKSetCandBreak( eCharBreak );

        if ( BRKExceedVals( adjustableSpace ) ) {
            BRKLineDecision( 0 );
            return BRKExitLoop( );
        }
        gbrS.cB.curBox  +=  gbrS.cB.theChRec ->escapement;

        gbrS.cB.chCount++;
        gbrS.cB.streamCount++;
        gbrS.cB.theChRec++;

        return in_line;
    }

    if ( gbrS.firstBox ) {
        adjustableSpace =  gbrS.desiredMeasure  -  gbrS.cB.curBox ;

            /* at the start of every word set a potential break point */
        BRKSetCandBreak(  eCharBreak );
        if ( BRKExceedVals(  adjustableSpace ) ) {
            BRKLineDecision( 0 );
            return BRKExitLoop(  );
        }

        BRKSetFirstBox(  );
    }

   gbrS.cB.curBox  +=  gbrS.cB.theChRec ->escapement;

   gbrS.cB.chCount++;
   gbrS.cB.streamCount++;
   gbrS.cB.theChRec++;

   return in_line;
}

/*************************************************************************/

static void getfield( stUnivString& ustr,
                      APPColumn     col,
                      scStream*     stream,
                      uint8         id,
                      TypeSpec&     spec )
{
    clField& field = clField::createField( stream, id );
    field.content( ustr, col, spec );
    field.release();
}

/* ===================================================================== */

static inline eBreakEvent bmBRKField()
{
    stUnivString    ustr;
    TypeSpec        spec = gbrS.cB.spec;

    getfield( ustr,
              gbrS.theBreakColH->GetAPPName(),
              gbrS.theBreakColH->GetStream(),
              gbrS.cB.theChRec->flags.GetField(),
              spec );

    if ( gbrS.firstBox ) {
        MicroPoint adjustableSpace = gbrS.desiredMeasure - gbrS.cB.curBox;

            /* at the start of every word set a potential break point */
        BRKSetCandBreak(  eCharBreak );
        gbrS.lineHyphenated = false;
        BRKRepairLastSpace( gbrS.cB.theChRec, gbrS.cB.trailingSpaces );
    }

    return gbrS.cB.theChRec->character;
}


/**********************************************************************/
/* we have tripped on a word space, if it is the first word space we
 * must do some housekeeping, the first word space test performs two
 * operations: 1. it tests to see if we have exceeded the measure and
 * 2. it counts actual interword spaces areas, we need to know that
 * for microjustification
 */

static inline eBreakEvent bmBRKWordSpace(  )
{
    BOOL bFirstGlue = gbrS.firstGlue;

    if ( gbrS.firstGlue ) {
        gbrS.firstBox = true;
        gbrS.firstGlue = false;
        gbrS.cB.wsSpaceCount++;
    }

    gbrS.fNoStartline = false;
    gbrS.fLastHangable = 0;

    gbrS.tmpOptGlue += scCachedStyle::GetCurrentCache().GetOptWord();
    gbrS.tmpMinGlue += scCachedStyle::GetCurrentCache().GetMinWord();
    gbrS.tmpMaxGlue += scCachedStyle::GetCurrentCache().GetMaxWord();

    gbrS.cB.theChRec->escapement = scCachedStyle::GetCurrentCache().GetOptWord();

    gbrS.cB.trailingSpaces++;
    gbrS.cB.spaceCount++;
    gbrS.cB.streamCount++;
    gbrS.cB.theChRec++;

    if (!bFirstGlue)
    {
        MicroPoint adjustableSpace = gbrS.desiredMeasure - gbrS.cB.curBox;
        BRKSetCandBreak( eCharBreak );
        if (BRKExceedVals (adjustableSpace))
        {
            BRKLineDecision (0);
            return BRKExitLoop( );
        }
    }

    return in_line;
}

/**********************************************************************/
/* does much the same as the word space break, except these are characters,
 * it also checks for hyphens
 */

static inline void BRKSetFirstBox(  )
{
     gbrS.firstGlue          = true;
     gbrS.firstBox           = false;
     gbrS.cB.minGlue         +=  gbrS.tmpMinGlue;
     gbrS.cB.optGlue         +=  gbrS.tmpOptGlue;
     gbrS.cB.maxGlue         +=  gbrS.tmpMaxGlue;
     gbrS.tmpMinGlue         =  gbrS.tmpOptGlue   =  gbrS.tmpMaxGlue   = 0;
     gbrS.cB.trailingSpaces = 0;
}

/**********************************************************************/

static inline eBreakEvent bmBRKChar(  )
```

```
    }
    else {
        lineData.fInkExtents.y1 += gbrS.minRelPosition;
        lineData.fInkExtents.y2 += ( lineData.fComputedLen - lastChWidth );
    }


#if SCDEBUG > 1
    SCDebugTrace( 4, scString( "BRKMaxLineVals: (%d %d %d %d)\n" ),
                  muPoints( lineData.fInkExtents.x1 ), muPoints( lineData.fInkExtents.y1 ),
                  muPoints( lineData.fInkExtents.x2 ), muPoints( lineData.fInkExtents.y2 ) );
#endif

    if ( gbrS.dcSet ) {
        /* now union the drop cap extents and the 'line' extents */
        lineData.fInkExtents.Union( dcRect );
    }
}

/*********************************************************************/

static inline Bool BRKExceedVals(  MicroPoint adjustableSpace )
{
    return gbrS.cB.minGlue + gbrS.tmpMinGlue > adjustableSpace;
}

/*********************************************************************/
/* this is the routine that effectively does the quality line breaking
 * up to this point we have simply been looking for a condition to
 * have been exceeded, now we may search to find a good break point
 */

static UCS2 BRKLineDecision( MicroPoint )
{
    CandBreak   *choice;

    if ( ( gbrS.effectiveRag & eRagFlag ) == eRagJustified )
        choice  = BRKHyphenateJust( );
    else
        choice  = BRKHyphenateRag( );

    if ( choice->breakCount == 0    &&
            gbrS.cB.breakCount > 1  &&
            choice->streamCount == choice->startCount )
        choice++;

    choice->spaceCount      = (ushort)(choice->spaceCount - choice->trailingSpaces);
    choice->wsSpaceCount    = (ushort)(choice->wsSpaceCount - choice->trailingSpaces);
    gbrS.cB = *choice;

    /* this is a fix for a bug in the character loop,
     * in the loop the spec is incremenented before the character is
     * called, if the character forces a line break, the spec at the
     * end of the line ( stored in choice->spec ) is invalid, the following
     * fixes this
     */

    while ( gbrS.cB.theChRec->character &&
            gbrS.cB.lineVal             &&
            gbrS.cB.specRec->offset() >= (long)gbrS.cB.streamCount ) {
        gbrS.cB.specRec--;
        gbrS.cB.spec = gbrS.cB.specRec->spec();
        gbrS.cB.lineVal--;
    }

    scCachedStyle::GetCachedStyle( gbrS.cB.spec );

    if ( gbrS.cB.breakVal == eHyphBreak ) {
        gbrS.lineHyphenated = true;
        if ( gbrS.cB.theChRec->flags.IsKernPresent() )
            gbrS.cB.curBox += BRKKernCorrection(  gbrS.cB.theChRec );
    }
    else {
```

```
    /* increment the line hyphenation count, used to prevent to many
     * consecutive lines hyphenated
     */
    if (  gbrS.lineHyphenated )
        linesHyphenated += 1;
    else
        linesHyphenated = 0;

    /* set these - the leading may force the line to be replaced & rebroken */
    BRKMaxLineVals( lineData, initialLead, initialBaseline, (gbrS.cB.theChRec-1)->character>=scWordS
pace?(gbrS.cB.theChRec-1)->escapement:0 );

    for ( ; (long)gbrS.cB.streamCount > (*specRec+1)->offset(); (*specRec)++ )
        ;

    if (  gbrS.foundCharIndent )
        BRKSetCharIndent(  chRec, startCount, count,  gbrS.letterSpaceAdj  );

    (gbrS.cB.theChRec-1)->flags.SetLineBreak();
    scAssert( count >= 0 );


#if _DEBUG
    eBreakType ret = gbrS.cB.breakVal;
    gbrS.Init();
    return ret;
#else
    return gbrS.cB.breakVal;
#endif

}

/************************************************************************/

static void BRKMaxLineVals( scLINERefData&      lineData,
                            MicroPoint      initialLead,
                            eFntBaseline    baseline,
                            MicroPoint      lastChWidth )
{
    scAngle     maxAngle    = lineData.fStartAngle;
    int         i;
    scXRect     dcRect;

    if (  gbrS.dcSet ) {
            /* set the max extents of the drop char before we bash linedata,
             * since the dc has to have the linedata that was in effect at
             * the beginning of the line
             */
        gbrS.dcInfo.dcMinY = MIN( gbrS.dcInfo.dcMinY, lineData.fOrg.y + lineData.fInkExtents.y1 );
        gbrS.dcInfo.dcMaxY = gbrS.dcInfo.dcMaxY - ( gbrS.dcInfo.dcMinY + lineData.fBaselineJump );
        gbrS.dcInfo.dcMaxX -= gbrS.dcInfo.dcMinX;

        dcRect.Set( gbrS.dcInfo.dcMinX, gbrS.dcInfo.dcMinY, gbrS.dcInfo.dcMaxX, gbrS.dcInfo.dcMaxY )
;
    }

    for ( i = gbrS.cB.lineVal; i--; ) {
        if ( initialLead < gbrS.fMaxLineVals[i].fMaxLead.GetLead() ) {
            if ( lineData.fEndLead.GetLead() < gbrS.fMaxLineVals[i].fMaxLead.GetLead() ) {
                lineData.fEndLead       = gbrS.fMaxLineVals[i].fMaxLead;
                lineData.SetMaxLeadSpec( gbrS.fMaxLineVals[i].fSpecRec->spec() );
            }
        }

        scXRect inkExtents( gbrS.fMaxLineVals[i].fMaxInkExtents );
        inkExtents.Translate( lineData.fOrg );
        lineData.fInkExtents.Union( inkExtents );
    }


    if ( lineData.IsHorizontal() ) {
        lineData.fInkExtents.x1 += gbrS.minRelPosition;
        lineData.fInkExtents.x2 += ( lineData.fComputedLen - lastChWidth );
    }


    if ( gbrS.cB.theChRec->character == scEndStream ) {
        if ( startCount==0 || (gbrS.cB.theChRec-1)->character!=scHardReturn ) {
            BRKPlaceLine( lineData.fOrg, lineData.fComputedLen, scCachedStyle::GetCurrentCache().Get
Flowdir() );
            count = gbrS.cB.streamCount - startCount;
            BRKMaxLineVals( lineData, initialLead, initialBaseline, 0 );
#if _DEBUG
    gbrS.Init();
#endif

            return eEndStreamBreak;
        }
    }

    switch ( BRKTheLoop() ) {
        case start_of_line:
            break;
        case end_of_stream_reached:
                /* the end of paragraph has been detected */
                /* force justify the line,
                 * NOTE: we should probably justify the line
                 * if it is close to the desired measure
                 * I THINK WE DO
                 */
            BRKRepairFinalSpace(  );
            lineData.fRagSetting = gbrS.effectiveRag;
            if ( gbrS.colShapeRag != (eTSJust)-1 )
                gbrS.effectiveRag = eRagRight;

            if ( (gbrS.effectiveRag & eRagFlag) == eRagJustified && !gbrS.cB.fillSpCount ) {
                if ( !(gbrS.effectiveRag & (int)eLastLineJust) )
                    lineData.fComputedLen = gbrS.cB.curBox + gbrS.cB.optGlue;
                else {
                    if ( gbrS.allowJustification )
                        BRKJustifyLine(   );
                    lineData.fComputedLen =  gbrS.desiredMeasure;
                }
                lineData.fOrg.x += gbrS.brkLeftMargin;
                if ( gHiliteSpaces )
                    lineData.fComputedLen += gbrS.totalTrailingSpace;
            }
            else
                BRKPlaceLine( lineData.fOrg, lineData.fComputedLen, scCachedStyle::GetCurrentCache()
.GetFlowdir() );
            count       = gbrS.cB.streamCount - startCount;
            letterSpace = gbrS.letterSpaceAdj;
            break;
        default:
        case measure_exceeded:
                // the line measure has been exceeded, there are still more
                // characters in the paragraph
            lineData.fRagSetting = gbrS.effectiveRag;
            if ( gbrS.colShapeRag != (eTSJust)-1 )
                gbrS.effectiveRag = eRagRight;

            if ( (gbrS.effectiveRag & eRagFlag) == eRagJustified && !gbrS.cB.fillSpCount ) {
                if ( gbrS.allowJustification )
                    BRKJustifyLine(  );
                lineData.fOrg.x += gbrS.brkLeftMargin;
                lineData.fComputedLen = gbrS.desiredMeasure;
                if ( gHiliteSpaces )
                    lineData.fComputedLen += gbrS.totalTrailingSpace;
            }
            else
                BRKPlaceLine( lineData.fOrg, lineData.fComputedLen, scCachedStyle::GetCurrentCache()
.GetFlowdir() );
            count       = gbrS.cB.streamCount - startCount;
            letterSpace = gbrS.letterSpaceAdj;
            break;
    }
```

```
            /* zero out the char indent at the start of every paragraph */
    if ( startCount == 0 )
        gbrS.charIndent = LONG_MIN;
    gbrS.foundCharIndent = false;

    gbrS.cB.startCount      = gbrS.cB.streamCount       = (long)startCount;
    gbrS.theSpecRec         = *specRec;
    gbrS.cB.spec            = gbrS.theSpecRec->spec();
    gbrS.originalMeasure    = lineData.fComputedLen = lineData.fMeasure;

    /* this test is somewhat arbitrary, but we are reaching the
     * outer limits of our unit system, and what we probably
     * have encountered is a horizontally flexible column,
     * which should not be justified!!
     */
    gbrS.allowJustification = ( gbrS.originalMeasure < (LONG_MAX-one_pica) );

    /* a little bullet proofing */
    if (  gbrS.originalMeasure < 0 )
         gbrS.originalMeasure = one_pica * 2;

    if ( lineData.IsHorizontal() )
        gbrS.colShapeRag        = (eTSJust)( lineData.fColShapeType & eHorzFlex ? eRagLeft : -1 );
    else
        gbrS.colShapeRag        = (eTSJust)( lineData.fColShapeType & eVertFlex ? eRagLeft : -1 );

    gbrS.lastLineLen        = lineData.fLastLineLen;

    gbrS.theLineCount       = lineCount;
    gbrS.cB.breakVal        = eCharBreak;
    gbrS.cB.lineVal         = 0;
    gbrS.cB.breakCount      = 0;
    gbrS.cB.spaceCount      = 0;
    gbrS.cB.trailingSpaces  = 0;
    gbrS.cB.wsSpaceCount    = 0;
    gbrS.cB.fillSpCount     = 0;
    gbrS.cB.chCount         = 0;

    gbrS.letterSpaceAdj     = 0;
    gbrS.minRelPosition     = 0;
    gbrS.cB.curBox          = 0;

    initialLead     = lineData.fInitialLead.GetLead();
    initialBaseline = scCachedStyle::GetCurrentCache().GetBaselineType( );

    gbrS.cB.optGlue         = gbrS.cB.minGlue = gbrS.cB.maxGlue = 0L;
    gbrS.tmpMinGlue         = gbrS.tmpOptGlue   = gbrS.tmpMaxGlue = 0L;

    gbrS.cB.specChanged     = false;
    gbrS.firstBox           = gbrS.firstGlue        = true;
    gbrS.fNoStartline       = false;
    gbrS.fLastHangable      = 0;
    gbrS.numTargetChars     = 0;
    gbrS.totalTrailingSpace = 0;


    gbrS.desiredMeasure = ::BRKRagControl( gbrS.cB.theChRec, lineData.fOrg.x, lineData.fOrg.y,
                                    lineData.fMeasure, gbrS.cB.spec, lineCount, linesHyphenat
ed );

    if (  gbrS.desiredMeasure <= 0 ) {
        if ( !gbrS.dcSet )
            count = 0;
        else {
            BRKPlaceLine( lineData.fOrg, lineData.fComputedLen, scCachedStyle::GetCurrentCache().Get
Flowdir() );
            count = gbrS.cB.streamCount - startCount;
            BRKMaxLineVals( lineData, initialLead, initialBaseline, 0 );
        }
#if _DEBUG
    gbrS.Init();
#endif
        return eCharBreak;
```

```
static void ShuffleBreakCandidates()
{
    gbrS.candBreak[0].Init();  // force the clearing of a spec

        // shuffle the array down into the spot we just cleared
    SCmemmove( gbrS.candBreak, gbrS.candBreak+1, sizeof(CandBreak) * (MAXBREAKVALS-1L) );

        // zero out the last entry which is now doubled because we copied it into
        // the next to last entry
    SCmemset( gbrS.candBreak + ( MAXBREAKVALS - 1 ), 0, sizeof(CandBreak) );

        // initialize the last entry
    gbrS.candBreak[MAXBREAKVALS-1].Init();
}

/*************************************************************************/
/* put the current break values into the candidate break array */

static void BRKSetCandBreak( eBreakType breakType )
{
    CandBreak           *theBreak   = gbrS.candBreak + gbrS.cB.breakCount;

    *theBreak           = gbrS.cB;
    theBreak->breakVal  = breakType;

    if ( breakType == eHyphBreak )
        theBreak->curBox += scCachedStyle::GetCurrentCache().GetEscapement( '-' );

    if ( gbrS.cB.breakCount >= (MAXBREAKVALS-1) )
        ShuffleBreakCandidates();
    else
        gbrS.cB.breakCount++;
}

/*************************************************************************/

static eBreakEvent BRKExitLoop(  )
{
    return measure_exceeded;
}

/*************************************************************************/
/* we are passed in some characters & line attributes, break the line,
 * performing justification, hyphenation, indents, etc.
 */

eBreakType
BRKRomanLineBreak( CharRecordP      chRec,      // the character array
                   long             startCount, // # into char array to start the linebreak
                   long&            count,      // count into char array of end of line
                   scLINERefData&   lineData,
                   short            lineCount,
                   short&           linesHyphenated,
                   scSpecRecord**   specRec,
                   scXRect&,
                   GlyphSize&       letterSpace )
{
    /* not a good idea to use register or auto variables that are used
     * across a call to setjmp/longjmp, they alter them when the previous
     * stack is restored
     */
    MicroPoint      initialLead;
    eFntBaseline    initialBaseline;

    gbrS.Init();

    /* set up state variables */
    gbrS.gStartRec = gbrS.cB.theChRec = chRec + startCount;

    gbrS.fMaxLineVals[0] = gbrS.fZeroMaxLineVals;

    lineData.fInkExtents.Translate( lineData.fOrg );
```

```
static void         BRKDropCapControl(  MicroPoint, MicroPoint );
static void         BRKPlaceLine( scMuPoint&, MicroPoint&, const scFlowDir& );
static void         BRKJustifyLine( void );
static void         BRKSpecial(  ushort );
static MicroPoint   BRKRagControl(  CharRecordP,
                                    MicroPoint, MicroPoint,
                                    MicroPoint, TypeSpec,   ushort, short );

static CandBreak*   BRKLineDecisionJust( void );
static CandBreak*   BRKLineDecisionRag( void );
static void         BRKSetCharIndent( CharRecordP, long, long, MicroPoint );

static CandBreak*   BRKHyphenateRag( void );
static CandBreak*   BRKHyphenateJust( void );

static void         BRKAddHyphens( CandBreak*, CandBreak* );
#ifdef scForceBreakFirstWord
static void         BRKForceHyphens( CandBreak*, CandBreak* );
#endif
static short        BRKPerformDiscHyphen( CharRecordP, CharRecordP, Hyphen* );

static void         BRKAdjustWordSpace( CharRecordP, GlyphSize, long, long );

static void         BRKRepairLastSpace(  CharRecordP, long );
static void         BRKRepairFinalSpace( void );
static CharRecordP  BRKLastCharOnLine( CharRecordP );
static MicroPoint   BRKHangPuncRightAdjust( void );

static TypeSpec     BRKUpdateSpec( scSpecRecord* );

static MicroPoint   BRKKernCorrection( CharRecordP );

static void         BRKMaxLineVals( scLINERefData&, MicroPoint, eFntBaseline, MicroPoint );

/* @@@@@@@@ */

/*************************************************************************/

CandBreak::CandBreak()
{
    Init();
}

/*************************************************************************/

void CandBreak::Init()
{
    breakCount      = 0;
    startCount      = 0;
    streamCount     = 0;
    wsSpaceCount    = 0;
    spaceCount      = 0;
    trailingSpaces  = 0;
    chCount         = 0;
    fillSpCount     = 0;
    lineVal         = 0;
    breakVal        = eUndefinedBreak;
    minGlue         = 0;
    optGlue         = 0;
    maxGlue         = 0;
    curBox          = 0;
    fHangable       = 0;
    theChRec        = 0;
    specChanged     = 0;
    spec.clear();
    specRec         = 0;
}

/*************************************************************************/
// if the break candidates fill we remove the first entry and shuffle the
// candidates down since the first candidate is no longer a real candidate
```

```
/***************************************************************************

    File:       SCBREAK.C


    $Header: /Projects/Toolbox/ct/Scbreak.cpp 6     5/30/97 8:45a Wmanis $

    Contains:   line breaker

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.


 **************************************************************************/

#include "scbreak.h"
#include "sccolumn.h"
#include "scglobda.h"
#include "scstcach.h"
#include "scctype.h"
#include "scmem.h"
#include "screfdat.h"
#include "sccallbk.h"

// TOOLBOX BEHAVIOR - force first word on line to break if it does not fit
#define scForceBreakFirstWord

#define MAXLEADVALS         50
#define MAXBREAKVALS        100

#ifndef LETTERSPACE
#define LETTERSPACE( ch ) ( (ch)->character!=scWordSpace\
                                   && ((ch)+1)->character!=scWordSpace )
#endif

/*****************************************************************/
/*****************************************************************/
/*lint -esym(534,BRKLineDecision) */

static MicroPoint   BRKNextTokenWidth( CharRecordP, UCS2 );
static Bool         TabBreakChar( UCS2 theCh, UCS2 breakCh );

static eBreakEvent      BRKLoopBody( void );
static eBreakEvent      BRKTheLoop( void );

static Bool         BRKStillMoreChars( CharRecordP, long );

static void         BRKCharJapanese( void );

static eBreakEvent      bmBRKWordSpace( void );
static eBreakEvent      bmBRKFixSpace( void );
static eBreakEvent      bmBRKRelSpace( void );
static eBreakEvent      bmBRKEndStream( void );
static eBreakEvent      bmBRKChar( void );
static eBreakEvent      bmBRKHardReturn( void );
static eBreakEvent      bmBRKQuad( void );
static eBreakEvent      bmBRKField( void );
static eBreakEvent      bmBRKVertTab( void );
static eBreakEvent      bmBRKTab( void );
static eBreakEvent      bmBRKFillSpace( void );
static eBreakEvent      bmBRKRule( void );
static eBreakEvent      bmBRKHyphen( void );
```

```
    #endif
#endif

#if defined( SubDiv256 )
#define scBezBlendSize    (SubDiv256 + 1)
#elif defined( SubDiv128 )
#define scBezBlendSize    (SubDiv128 + 1)
#elif defined( SubDiv64 )
#define scBezBlendSize    (SubDiv64 + 1)
#elif defined( SubDiv32 )
#define scBezBlendSize    (SubDiv32 + 1)
#elif defined( SubDiv16 )
#define scBezBlendSize    (SubDiv16 + 1)
#elif defined( SubDiv8 )
#define scBezBlendSize    (SubDiv8 + 1)
#elif defined( SubDiv4 )
#define scBezBlendSize    (SubDiv4 + 1)
#elif defined( SubDiv2 )
#define scBezBlendSize    (SubDiv2 + 1)
#elif !defined( scBezBlendSize )
#define scBezBlendSize    0
#endif

#endif /* _H_SCBEZIER */
```

09629370 . 073100

```
/***************************************************************************

	File:		SCBEZIER.H

	$Header: /Projects/Toolbox/ct/SCBEZIER.H 2     5/30/97 8:44a Wmanis $

	Contains:	size of bezier sub-division factors

	Written by: Manis

	Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
	All rights reserved.

	This notice is intended as a precaution against inadvertent publication
	and does not constitute an admission or acknowledgment that publication
	has occurred or constitute a waiver of confidentiality.

	Composition Toolbox software is the proprietary
	and confidential property of Stonehand Inc.

***************************************************************************/

#ifndef _H_SCBEZIER
#define _H_SCBEZIER

#include "sctypes.h"

void			BEZVectorizePoly( scVertex *&, const scVertex * );


/* last one of these defined determines the number of vectors
 * that will be created by sub-dividing the bezier curver
 */
#define SubDiv2			2
#define SubDiv4			(SubDiv2 * 2)
#define SubDiv8			(SubDiv4 * 2)
#define SubDiv16		(SubDiv8 * 2)

#if 0
#define SubDiv32		(SubDiv16 * 2)
#define SubDiv64		(SubDiv32 * 2)
#define SubDiv128		(SubDiv64 * 2)
#define SubDiv256		(SubDiv128 * 2)
#endif


//#define scBezFixed
#define scBezREAL

#ifdef scBezFixed
	#define scBezFactor( x, y ) x
	struct scBezBlendValue {
		ushort	ca;
		ushort	cb;
		ushort	cc;
		ushort	cd;
	};
	#ifdef scBezREAL
		#error "can't define both"
	#endif
#endif

#ifdef scBezREAL
	#define scBezFactor( x, y ) y
	struct scBezBlendValue {
		REAL	ca;
		REAL	cb;
		REAL	cc;
		REAL	cd;
	};
	#ifdef scBezFixed
		#error "can't define both"
```

```
#endif

/* ================================================================== */

#ifdef scBezREAL

static void BezCompute( scVertex*       dstV,
                        const scVertex* srcV )
{
    int                 i;

    dstV[0]                 = srcV[0];
    dstV[scBezBlendSize-1]  = srcV[3];


    for ( i = 1;  i < scBezBlendSize - 1;  i++ ) {
        dstV[i].x = scRoundMP( srcV[0].x * bezblend[i].ca +
                               srcV[1].x * bezblend[i].cb +
                               srcV[2].x * bezblend[i].cc +
                               srcV[3].x * bezblend[i].cd );

        dstV[i].y = scRoundMP( srcV[0].y * bezblend[i].ca +
                               srcV[1].y * bezblend[i].cb +
                               srcV[2].y * bezblend[i].cc +
                               srcV[3].y * bezblend[i].cd );
        dstV[i].fPointType  = eCornerPoint;
    }
}

#endif

/* ================================================================== */
/* ================================================================== */


SCBezVertex *SCBezCompDrawList( SCBezVertex*    theVerts,
                                SCBezVertex*    drawList )
{
    SCBezVertex*        pDraw;
    scBezBlendValue*    pBlend;
    short               i;

    drawList[0]                 = theVerts[0];
    drawList[scBezBlendSize-1]  = theVerts[3];

    pBlend  = bezblend + 1;
    pDraw   = drawList + 1;

    for (i = 0;  i < scBezBlendSize-2;  i++) {
        pDraw->x =(short)(((long)theVerts[0].x * (ulong)pBlend->ca
                + (long)theVerts[1].x * (ulong)pBlend->cb
                + (long)theVerts[2].x * (ulong)pBlend->cc
                + (long)theVerts[3].x * (ulong)pBlend->cd) >> 16);
        pDraw->y =(short)(((long)theVerts[0].y * (ulong)pBlend->ca
                + (long)theVerts[1].y * (ulong)pBlend->cb
                + (long)theVerts[2].y * (ulong)pBlend->cc
                + (long)theVerts[3].y * (ulong)pBlend->cd) >> 16);
        pBlend++;
        pDraw++;
    }
    return drawList;
}

/* ================================================================== */

static void BezCompute( SCVertex*       dstV,
                        const SCVertex* srcV )
{
    SCBezVertex     v[4];
    SCBezVertex     drawList[scBezBlendSize];
    register int    i;
    scPointType     fPointType;
    short           minX;
    short           minY;

    pointType = srcV->fPointType;

    /* put source into a 16 bit quantity so that
     * we can perform fixed point multiplies on it,
     * and force bezier into positive coordinate space
     * so that the fixed point multiplies with blending
     * values will work
     */

    minX = minY = SHRT_MAX;
    for ( i = 0; i < 4; i++ ) {
        v[i].x  = (short)( srcV->x >> 16 );
        minX    = MIN( minX, v[i].x );
        v[i].y  = (short)(srcV->y >> 16);
        minY    = MIN( minY, v[i].y );
        srcV++;
    }

    minX = ( minX < 0 ? -minX : 0 );
    minY = ( minY < 0 ? -minY : 0 );
    if ( minX || minY ) {
        for ( i = 0; i < 4; i++ ) {
            v[i].x += minX;
            v[i].y += minY;
        }
    }
    SCBezCompDrawList( v, drawList ) ;
    RenderBezier( dstV, pointType, drawList, minX, minY );
}
```

```
void BEZVectorizePoly( scVertex*&       dstV,
                       const scVertex*  srcV )
{
    scVertex*   vList;
    int         i;
    Bool        process;
    long        numPoints;
    long        numCurves;
    scVertex    bezVectors[ scBezBlendSize + 2 ];

    numPoints = CountVerts( srcV );
    numCurves = CountBezCurves( srcV );

    if ( numCurves == 0 )
        return;

    long     segments = numPoints + scBezBlendSize * numCurves;
    dstV = vList = (scVertex*)MEMAllocPtr( segments * sizeof( scVertex ) );

    for ( process = true; process; ) {

        switch ( srcV->fPointType ) {

            case eFinalPoint:
                process = false;
            default:
                *vList++ = *srcV++;
                break;

            case eBezControlPoint:
                    // insure we have two points - if not process normally
                if ( (srcV+1)->fPointType == eBezControlPoint ) {
                    BezCompute( bezVectors, srcV - 1 );
                    for ( i = 1; i < scBezBlendSize - 1; i++ )
                        *vList++ = bezVectors[i];
                    srcV += 2;
                }
                else
                    *vList++ = *srcV++;
                break;
        }
    }
}
/* ================================================================== */

#ifdef scBezFixed

/* it is assumed that the list has enough space before entering
 * this routine
 */

static void RenderBezier( SCVertex*      dstV,
                          scPointType    type,
                          SCBezVertex*   draw,
                          short          minX,
                          short          minY )
{
    size_t  i;

    for ( i = 0; i < scBezBlendSize; i++, draw++ ) {
        if ( i == 0 )
            dstV->fPointType = type;
        else
            dstV->fPointType = eCornerPoint;

        dstV->x = (long)( draw->x - minX ) << 16;
        dstV->y = (long)( draw->y - minY ) << 16;
        dstV++;
    }
}
```

```
/***************************************************************************

    File:       SCBEZIER.C


    $Header: /Projects/Toolbox/ct/SCBEZIER.CPP 2     5/30/97 8:45a Wmanis $

    Contains:       vectorizes beziers

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

***************************************************************************/

#include "scbezier.h"
#include "scmem.h"
#include <limits.h>

struct SCBezVertex {
    short   x;
    short   y;
};

/* ================================================================== */

extern scBezBlendValue bezblend[];

static void BezCompute( scVertex*       dstV,
                        const scVertex* srcV );


/* ================================================================== */
/* count the number of vertices in the vertex list */

inline long CountVerts( const scVertex* verts )
{
    long    numVerts;

    for ( numVerts = 1; verts->fPointType != eFinalPoint; verts++, numVerts++ )
        ;
    return numVerts;
}


/* ================================================================== */
/* count the number of bezier curves in a vertex list */

inline long CountBezCurves( const scVertex* verts )
{
    long    numCurves;

    for ( numCurves = 0; verts->fPointType != eFinalPoint; ) {
        if ( verts->fPointType == eBezControlPoint && (verts+1)->fPointType == eBezControlPoint ) {
            numCurves++;
            verts += 2;
        }
        else
            verts++;
    }
    return numCurves;
}

/* ================================================================== */
/* process the list vectorizing the beziers into straight lines */
```

```
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x0000,     0.0000016 ),   /*  (253 0) */
    scBezFactor( 0x001a,     0.0004072 ),   /*  (253 1) */
    scBezFactor( 0x08ca,     0.0343371 ),   /*  (253 2) */
    scBezFactor( 0xf71a,     0.9652541 )    /*  (253 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x0000,     0.0000005 ),   /*  (254 0) */
    scBezFactor( 0x000b,     0.0001817 ),   /*  (254 1) */
    scBezFactor( 0x05e8,     0.0230727 ),   /*  (254 2) */
    scBezFactor( 0xfa0b,     0.9767451 )    /*  (254 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x0000,     0.0000001 ),   /*  (255 0) */
    scBezFactor( 0x0002,     0.0000456 ),   /*  (255 1) */
    scBezFactor( 0x02fa,     0.0116274 ),   /*  (255 2) */
    scBezFactor( 0xfd02,     0.9883270 )    /*  (255 3) */
},
#endif


/* this one is needed by all sub divisions */
{
    scBezFactor( 0x0000,     0.0000000 ),   /*  (256 0) */
    scBezFactor( 0x0000,     0.0000000 ),   /*  (256 1) */
    scBezFactor( 0x0000,     0.0000000 ),   /*  (256 2) */
    scBezFactor( 0x0000,     1.0000000 )    /*  (256 3) */
}

};
```

```
    scBezFactor( 0x1e39,     0.1180664 ),   /*  (245 2) */
    scBezFactor( 0xe065,     0.8765534 )    /*  (245 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x0003,     0.0000596 ),   /*  (246 0) */
    scBezFactor( 0x0120,     0.0043988 ),   /*  (246 1) */
    scBezFactor( 0x1bb3,     0.1082110 ),   /*  (246 2) */
    scBezFactor( 0xe328,     0.8873305 )    /*  (246 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x0002,     0.0000435 ),   /*  (247 0) */
    scBezFactor( 0x00ea,     0.0035775 ),   /*  (247 1) */
    scBezFactor( 0x1922,     0.0981833 ),   /*  (247 2) */
    scBezFactor( 0xe5f0,     0.8981957 )    /*  (247 3) */
},
#endif


#ifdef SubDiv32
{
    scBezFactor( 0x0002,     0.0000305 ),   /*  (248 0) */
    scBezFactor( 0x00ba,     0.0028381 ),   /*  (248 1) */
    scBezFactor( 0x1686,     0.0879822 ),   /*  (248 2) */
    scBezFactor( 0xe8be,     0.9091492 )    /*  (248 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x0001,     0.0000204 ),   /*  (249 0) */
    scBezFactor( 0x008e,     0.0021817 ),   /*  (249 1) */
    scBezFactor( 0x13de,     0.0776065 ),   /*  (249 2) */
    scBezFactor( 0xeb91,     0.9201913 )    /*  (249 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x0000,     0.0000129 ),   /*  (250 0) */
    scBezFactor( 0x0069,     0.0016093 ),   /*  (250 1) */
    scBezFactor( 0x112a,     0.0670552 ),   /*  (250 2) */
    scBezFactor( 0xee6b,     0.9313226 )    /*  (250 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x0000,     0.0000075 ),   /*  (251 0) */
    scBezFactor( 0x0049,     0.0011221 ),   /*  (251 1) */
    scBezFactor( 0x0e6b,     0.0563273 ),   /*  (251 2) */
    scBezFactor( 0xf14a,     0.9425432 )    /*  (251 3) */
},
#endif


#ifdef SubDiv64
{
    scBezFactor( 0x0000,     0.0000038 ),   /*  (252 0) */
    scBezFactor( 0x002f,     0.0007210 ),   /*  (252 1) */
    scBezFactor( 0x0ba0,     0.0454216 ),   /*  (252 2) */
    scBezFactor( 0xf42f,     0.9538536 )    /*  (252 3) */
},
{
    scBezFactor( 0x0016,     0.0003476 ),   /*  (238 0) */
    scBezFactor( 0x0387,     0.0137887 ),   /*  (238 1) */
    scBezFactor( 0x2eac,     0.1823173 ),   /*  (238 2) */
    scBezFactor( 0xcdb5,     0.8035464 )    /*  (238 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x0013,     0.0002928 ),   /*  (239 0) */
    scBezFactor( 0x0329,     0.0123509 ),   /*  (239 1) */
    scBezFactor( 0x2c73,     0.1736385 ),   /*  (239 2) */
    scBezFactor( 0xd04f,     0.8137178 )    /*  (239 3) */
},
#endif


#ifdef SubDiv16
{
    scBezFactor( 0x0010,     0.0002441 ),   /*  (240 0) */
    scBezFactor( 0x02d0,     0.0109863 ),   /*  (240 1) */
    scBezFactor( 0x2a30,     0.1647949 ),   /*  (240 2) */
    scBezFactor( 0xd2f0,     0.8239746 )    /*  (240 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x000d,     0.0002012 ),   /*  (241 0) */
    scBezFactor( 0x027b,     0.0096962 ),   /*  (241 1) */
    scBezFactor( 0x27e1,     0.1557854 ),   /*  (241 2) */
    scBezFactor( 0xd595,     0.8343173 )    /*  (241 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x000a,     0.0001636 ),   /*  (242 0) */
    scBezFactor( 0x022b,     0.0084815 ),   /*  (242 1) */
    scBezFactor( 0x2588,     0.1466088 ),   /*  (242 2) */
    scBezFactor( 0xd841,     0.8447461 )    /*  (242 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x0008,     0.0001310 ),   /*  (243 0) */
    scBezFactor( 0x01e1,     0.0073434 ),   /*  (243 1) */
    scBezFactor( 0x2323,     0.1372642 ),   /*  (243 2) */
    scBezFactor( 0xdaf2,     0.8552615 )    /*  (243 3) */
},
#endif


#ifdef SubDiv64
{
    scBezFactor( 0x0006,     0.0001030 ),   /*  (244 0) */
    scBezFactor( 0x019b,     0.0062828 ),   /*  (244 1) */
    scBezFactor( 0x20b4,     0.1277504 ),   /*  (244 2) */
    scBezFactor( 0xdda9,     0.8658638 )    /*  (244 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x0005,     0.0000793 ),   /*  (245 0) */
    scBezFactor( 0x015b,     0.0053009 ),   /*  (245 1) */
```

```
#ifdef SubDiv256
{
    scBezFactor( 0x003d,     0.0009313 ),    /*  (231 0) */
    scBezFactor( 0x069b,     0.0258163 ),    /*  (231 1) */
    scBezFactor( 0x3d11,     0.2385423 ),    /*  (231 2) */
    scBezFactor( 0xbc15,     0.7347102 )     /*  (231 3) */
},
#endif


#ifdef SubDiv32
{
    scBezFactor( 0x0036,     0.0008240 ),    /*  (232 0) */
    scBezFactor( 0x061e,     0.0238953 ),    /*  (232 1) */
    scBezFactor( 0x3b22,     0.2309875 ),    /*  (232 2) */
    scBezFactor( 0xbe8a,     0.7442932 )     /*  (232 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x002f,     0.0007252 ),    /*  (233 0) */
    scBezFactor( 0x05a4,     0.0220401 ),    /*  (233 1) */
    scBezFactor( 0x3928,     0.2232755 ),    /*  (233 2) */
    scBezFactor( 0xc103,     0.7539592 )     /*  (233 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x0029,     0.0006347 ),    /*  (234 0) */
    scBezFactor( 0x052f,     0.0202518 ),    /*  (234 1) */
    scBezFactor( 0x3724,     0.2154050 ),    /*  (234 2) */
    scBezFactor( 0xc382,     0.7637086 )     /*  (234 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x0024,     0.0005520 ),    /*  (235 0) */
    scBezFactor( 0x04be,     0.0185314 ),    /*  (235 1) */
    scBezFactor( 0x3516,     0.2073750 ),    /*  (235 2) */
    scBezFactor( 0xc606,     0.7735416 )     /*  (235 3) */
},
#endif


#ifdef SubDiv64
{
    scBezFactor( 0x001f,     0.0004768 ),    /*  (236 0) */
    scBezFactor( 0x0452,     0.0168800 ),    /*  (236 1) */
    scBezFactor( 0x32fd,     0.1991844 ),    /*  (236 2) */
    scBezFactor( 0xc890,     0.7834587 )     /*  (236 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x001a,     0.0004088 ),    /*  (237 0) */
    scBezFactor( 0x03ea,     0.0152988 ),    /*  (237 1) */
    scBezFactor( 0x30da,     0.1908322 ),    /*  (237 2) */
    scBezFactor( 0xcb20,     0.7934602 )     /*  (237 3) */
},
#endif


#ifdef SubDiv128
```

```
    scBezFactor( 0xa936,     0.6609897 )    /*  (223 3) */
},
#endif


#ifdef SubDiv8
{
    scBezFactor( 0x0080,     0.0019531 ),   /*  (224 0) */
    scBezFactor( 0x0a80,     0.0410156 ),   /*  (224 1) */
    scBezFactor( 0x4980,     0.2871094 ),   /*  (224 2) */
    scBezFactor( 0xab80,     0.6699219 )    /*  (224 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x0074,     0.0017757 ),   /*  (225 0) */
    scBezFactor( 0x09e5,     0.0386640 ),   /*  (225 1) */
    scBezFactor( 0x47d7,     0.2806261 ),   /*  (225 2) */
    scBezFactor( 0xadce,     0.6789342 )    /*  (225 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x0069,     0.0016093 ),   /*  (226 0) */
    scBezFactor( 0x094f,     0.0363708 ),   /*  (226 1) */
    scBezFactor( 0x4624,     0.2739930 ),   /*  (226 2) */
    scBezFactor( 0xb022,     0.6880269 )    /*  (226 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x005f,     0.0014537 ),   /*  (227 0) */
    scBezFactor( 0x08bd,     0.0341368 ),   /*  (227 1) */
    scBezFactor( 0x4467,     0.2672090 ),   /*  (227 2) */
    scBezFactor( 0xb27b,     0.6972005 )    /*  (227 3) */
},
#endif


#ifdef SubDiv64
{
    scBezFactor( 0x0055,     0.0013084 ),   /*  (228 0) */
    scBezFactor( 0x082e,     0.0319633 ),   /*  (228 1) */
    scBezFactor( 0x42a1,     0.2602730 ),   /*  (228 2) */
    scBezFactor( 0xb4da,     0.7064552 )    /*  (228 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x004c,     0.0011732 ),   /*  (229 0) */
    scBezFactor( 0x07a4,     0.0298514 ),   /*  (229 1) */
    scBezFactor( 0x40d0,     0.2531839 ),   /*  (229 2) */
    scBezFactor( 0xb73e,     0.7157915 )    /*  (229 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x0044,     0.0010476 ),   /*  (230 0) */
    scBezFactor( 0x071e,     0.0278020 ),   /*  (230 1) */
    scBezFactor( 0x3ef5,     0.2459407 ),   /*  (230 2) */
    scBezFactor( 0xb9a7,     0.7252097 )    /*  (230 3) */
},
#endif
```

```
    scBezFactor( 0x00fa,     0.0038147 ),   /*  (216 0) */
    scBezFactor( 0x0fd2,     0.0617981 ),   /*  (216 1) */
    scBezFactor( 0x556e,     0.3337097 ),   /*  (216 2) */
    scBezFactor( 0x99c6,     0.6006775 )    /*  (216 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x00e7,     0.0035357 ),   /*  (217 0) */
    scBezFactor( 0x0f1b,     0.0590188 ),   /*  (217 1) */
    scBezFactor( 0x5411,     0.3283866 ),   /*  (217 2) */
    scBezFactor( 0x9beb,     0.6090589 )    /*  (217 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x00d6,     0.0032706 ),   /*  (218 0) */
    scBezFactor( 0x0e68,     0.0562892 ),   /*  (218 1) */
    scBezFactor( 0x52ab,     0.3229222 ),   /*  (218 2) */
    scBezFactor( 0x9e15,     0.6175179 )    /*  (218 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x00c5,     0.0030192 ),   /*  (219 0) */
    scBezFactor( 0x0db9,     0.0536104 ),   /*  (219 1) */
    scBezFactor( 0x513b,     0.3173155 ),   /*  (219 2) */
    scBezFactor( 0xa045,     0.6260549 )    /*  (219 3) */
},
#endif


#ifdef SubDiv64
{
    scBezFactor( 0x00b6,     0.0027809 ),   /*  (220 0) */
    scBezFactor( 0x0d0d,     0.0509834 ),   /*  (220 1) */
    scBezFactor( 0x4fc2,     0.3115654 ),   /*  (220 2) */
    scBezFactor( 0xa279,     0.6346703 )    /*  (220 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x00a7,     0.0025555 ),   /*  (221 0) */
    scBezFactor( 0x0c64,     0.0484094 ),   /*  (221 1) */
    scBezFactor( 0x4e40,     0.3056708 ),   /*  (221 2) */
    scBezFactor( 0xa4b3,     0.6433643 )    /*  (221 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x0099,     0.0023427 ),   /*  (222 0) */
    scBezFactor( 0x0bbf,     0.0458894 ),   /*  (222 1) */
    scBezFactor( 0x4cb4,     0.2996306 ),   /*  (222 2) */
    scBezFactor( 0xa6f2,     0.6521373 )    /*  (222 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x008c,     0.0021420 ),   /*  (223 0) */
    scBezFactor( 0x0b1d,     0.0434244 ),   /*  (223 1) */
    scBezFactor( 0x4b1f,     0.2934439 ),   /*  (223 2) */
```

```
#ifdef SubDiv256
{
    scBezFactor( 0x0195,     0.0061883 ),   /*  (209 0) */
    scBezFactor( 0x1522,     0.0825550 ),   /*  (209 1) */
    scBezFactor( 0x5dfa,     0.3671063 ),   /*  (209 2) */
    scBezFactor( 0x8b4d,     0.5441504 )    /*  (209 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x017c,     0.0058017 ),   /*  (210 0) */
    scBezFactor( 0x1457,     0.0794578 ),   /*  (210 1) */
    scBezFactor( 0x5cdc,     0.3627419 ),   /*  (210 2) */
    scBezFactor( 0x8d4f,     0.5519986 )    /*  (210 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x0163,     0.0054315 ),   /*  (211 0) */
    scBezFactor( 0x138f,     0.0764027 ),   /*  (211 1) */
    scBezFactor( 0x5bb5,     0.3582439 ),   /*  (211 2) */
    scBezFactor( 0x8f57,     0.5599219 )    /*  (211 3) */
},
#endif


#ifdef SubDiv64
{
    scBezFactor( 0x014c,     0.0050774 ),   /*  (212 0) */
    scBezFactor( 0x12c9,     0.0733910 ),   /*  (212 1) */
    scBezFactor( 0x5a86,     0.3536110 ),   /*  (212 2) */
    scBezFactor( 0x9163,     0.5679207 )    /*  (212 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x0136,     0.0047390 ),   /*  (213 0) */
    scBezFactor( 0x1207,     0.0704235 ),   /*  (213 1) */
    scBezFactor( 0x594d,     0.3488422 ),   /*  (213 2) */
    scBezFactor( 0x9374,     0.5759953 )    /*  (213 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x0121,     0.0044160 ),   /*  (214 0) */
    scBezFactor( 0x1147,     0.0675015 ),   /*  (214 1) */
    scBezFactor( 0x580c,     0.3439364 ),   /*  (214 2) */
    scBezFactor( 0x958a,     0.5841460 )    /*  (214 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x010d,     0.0041080 ),   /*  (215 0) */
    scBezFactor( 0x108b,     0.0646260 ),   /*  (215 1) */
    scBezFactor( 0x56c1,     0.3388926 ),   /*  (215 2) */
    scBezFactor( 0x97a5,     0.5923733 )    /*  (215 3) */
},
#endif


#ifdef SubDiv32
{
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x0267,     0.0093856 ),   /*  (202 0) */
    scBezFactor( 0x1af6,     0.1053271 ),   /*  (202 1) */
    scBezFactor( 0x64dd,     0.3940015 ),   /*  (202 2) */
    scBezFactor( 0x7dc4,     0.4912858 )    /*  (202 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x0245,     0.0088738 ),   /*  (203 0) */
    scBezFactor( 0x1a1a,     0.1019645 ),   /*  (203 1) */
    scBezFactor( 0x63fa,     0.3905434 ),   /*  (203 2) */
    scBezFactor( 0x7fa5,     0.4986183 )    /*  (203 3) */
},
#endif


#ifdef SubDiv64
{
    scBezFactor( 0x0225,     0.0083809 ),   /*  (204 0) */
    scBezFactor( 0x1940,     0.0986366 ),   /*  (204 1) */
    scBezFactor( 0x630f,     0.3869591 ),   /*  (204 2) */
    scBezFactor( 0x818a,     0.5060234 )    /*  (204 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x0206,     0.0079066 ),   /*  (205 0) */
    scBezFactor( 0x1868,     0.0953445 ),   /*  (205 1) */
    scBezFactor( 0x621c,     0.3832474 ),   /*  (205 2) */
    scBezFactor( 0x8374,     0.5135015 )    /*  (205 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x01e8,     0.0074506 ),   /*  (206 0) */
    scBezFactor( 0x1793,     0.0920892 ),   /*  (206 1) */
    scBezFactor( 0x6120,     0.3794074 ),   /*  (206 2) */
    scBezFactor( 0x8563,     0.5210528 )    /*  (206 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x01cb,     0.0070124 ),   /*  (207 0) */
    scBezFactor( 0x16c0,     0.0888718 ),   /*  (207 1) */
    scBezFactor( 0x601c,     0.3754379 ),   /*  (207 2) */
    scBezFactor( 0x8757,     0.5286779 )    /*  (207 3) */
},
#endif


#ifdef SubDiv16
{
    scBezFactor( 0x01b0,     0.0065918 ),   /*  (208 0) */
    scBezFactor( 0x15f0,     0.0856934 ),   /*  (208 1) */
    scBezFactor( 0x5f10,     0.3713379 ),   /*  (208 2) */
    scBezFactor( 0x8950,     0.5363770 )    /*  (208 3) */
},
#endif
```

```
    scBezFactor( 0x2223,     0.1333480 ),   /*  (194 1) */
    scBezFactor( 0x6ad0,     0.4172502 ),   /*  (194 2) */
    scBezFactor( 0x6f69,     0.4351964 )    /*  (194 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x0376,     0.0135291 ),   /*  (195 0) */
    scBezFactor( 0x2137,     0.1297465 ),   /*  (195 1) */
    scBezFactor( 0x6a2d,     0.4147634 ),   /*  (195 2) */
    scBezFactor( 0x7124,     0.4419610 )    /*  (195 3) */
},
#endif


#ifdef SubDiv64
{
    scBezFactor( 0x034b,     0.0128746 ),   /*  (196 0) */
    scBezFactor( 0x204c,     0.1261711 ),   /*  (196 1) */
    scBezFactor( 0x6983,     0.4121590 ),   /*  (196 2) */
    scBezFactor( 0x72e4,     0.4487953 )    /*  (196 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x0322,     0.0122415 ),   /*  (197 0) */
    scBezFactor( 0x1f64,     0.1226229 ),   /*  (197 1) */
    scBezFactor( 0x68d0,     0.4094358 ),   /*  (197 2) */
    scBezFactor( 0x74a8,     0.4556997 )    /*  (197 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x02fa,     0.0116296 ),   /*  (198 0) */
    scBezFactor( 0x1e7d,     0.1191030 ),   /*  (198 1) */
    scBezFactor( 0x6816,     0.4065928 ),   /*  (198 2) */
    scBezFactor( 0x7671,     0.4626746 )    /*  (198 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x02d3,     0.0110384 ),   /*  (199 0) */
    scBezFactor( 0x1d98,     0.1156123 ),   /*  (199 1) */
    scBezFactor( 0x6754,     0.4036290 ),   /*  (199 2) */
    scBezFactor( 0x783f,     0.4697203 )    /*  (199 3) */
},
#endif


#ifdef SubDiv32
{
    scBezFactor( 0x02ae,     0.0104675 ),   /*  (200 0) */
    scBezFactor( 0x1cb6,     0.1121521 ),   /*  (200 1) */
    scBezFactor( 0x668a,     0.4005432 ),   /*  (200 2) */
    scBezFactor( 0x7a12,     0.4768372 )    /*  (200 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x0289,     0.0099167 ),   /*  (201 0) */
    scBezFactor( 0x1bd5,     0.1087233 ),   /*  (201 1) */
    scBezFactor( 0x65b7,     0.3973344 ),   /*  (201 2) */
    scBezFactor( 0x7be9,     0.4840255 )    /*  (201 3) */
```

```
#ifdef SubDiv256
{
    scBezFactor( 0x0503,     0.0195807 ),   /*  (187 0) */
    scBezFactor( 0x28c1,     0.1591993 ),   /*  (187 1) */
    scBezFactor( 0x6e73,     0.4314532 ),   /*  (187 2) */
    scBezFactor( 0x63c7,     0.3897669 )    /*  (187 3) */
},
#endif


#ifdef SubDiv64
{
    scBezFactor( 0x04cc,     0.0187416 ),   /*  (188 0) */
    scBezFactor( 0x27cb,     0.1554451 ),   /*  (188 1) */
    scBezFactor( 0x6e04,     0.4297600 ),   /*  (188 2) */
    scBezFactor( 0x6563,     0.3960533 )    /*  (188 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x0496,     0.0179269 ),   /*  (189 0) */
    scBezFactor( 0x26d6,     0.1517095 ),   /*  (189 1) */
    scBezFactor( 0x6d8e,     0.4279566 ),   /*  (189 2) */
    scBezFactor( 0x6704,     0.4024070 )    /*  (189 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x0463,     0.0171361 ),   /*  (190 0) */
    scBezFactor( 0x25e2,     0.1479936 ),   /*  (190 1) */
    scBezFactor( 0x6d11,     0.4260421 ),   /*  (190 2) */
    scBezFactor( 0x68a8,     0.4088283 )    /*  (190 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x0430,     0.0163689 ),   /*  (191 0) */
    scBezFactor( 0x24f0,     0.1442984 ),   /*  (191 1) */
    scBezFactor( 0x6c8c,     0.4240152 ),   /*  (191 2) */
    scBezFactor( 0x6a52,     0.4153175 )    /*  (191 3) */
},
#endif


#ifdef SubDiv4
{
    scBezFactor( 0x0400,     0.0156250 ),   /*  (192 0) */
    scBezFactor( 0x2400,     0.1406250 ),   /*  (192 1) */
    scBezFactor( 0x6c00,     0.4218750 ),   /*  (192 2) */
    scBezFactor( 0x6c00,     0.4218750 )    /*  (192 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x03d0,     0.0149040 ),   /*  (193 0) */
    scBezFactor( 0x2310,     0.1369745 ),   /*  (193 1) */
    scBezFactor( 0x6b6c,     0.4196203 ),   /*  (193 2) */
    scBezFactor( 0x6db2,     0.4285012 )    /*  (193 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x03a2,     0.0142055 ),   /*  (194 0) */
```

```
#endif


#ifdef SubDiv64
{
    scBezFactor( 0x06b2,     0.0261650 ),   /*  (180 0) */
    scBezFactor( 0x2f97,     0.1859093 ),   /*  (180 1) */
    scBezFactor( 0x70b8,     0.4403114 ),   /*  (180 2) */
    scBezFactor( 0x58fd,     0.3476143 )    /*  (180 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x066f,     0.0251457 ),   /*  (181 0) */
    scBezFactor( 0x2e9b,     0.1820549 ),   /*  (181 1) */
    scBezFactor( 0x7079,     0.4393592 ),   /*  (181 2) */
    scBezFactor( 0x5a7b,     0.3534401 )    /*  (181 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x062e,     0.0241532 ),   /*  (182 0) */
    scBezFactor( 0x2d9f,     0.1782117 ),   /*  (182 1) */
    scBezFactor( 0x7034,     0.4383044 ),   /*  (182 2) */
    scBezFactor( 0x5bfd,     0.3593307 )    /*  (182 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x05ef,     0.0231872 ),   /*  (183 0) */
    scBezFactor( 0x2ca4,     0.1743806 ),   /*  (183 1) */
    scBezFactor( 0x6fe8,     0.4371459 ),   /*  (183 2) */
    scBezFactor( 0x5d83,     0.3652863 )    /*  (183 3) */
},
#endif


#ifdef SubDiv32
{
    scBezFactor( 0x05b2,     0.0222473 ),   /*  (184 0) */
    scBezFactor( 0x2baa,     0.1705627 ),   /*  (184 1) */
    scBezFactor( 0x6f96,     0.4358826 ),   /*  (184 2) */
    scBezFactor( 0x5f0e,     0.3713074 )    /*  (184 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x0576,     0.0213332 ),   /*  (185 0) */
    scBezFactor( 0x2ab0,     0.1667592 ),   /*  (185 1) */
    scBezFactor( 0x6f3c,     0.4345134 ),   /*  (185 2) */
    scBezFactor( 0x609c,     0.3773943 )    /*  (185 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x053b,     0.0204444 ),   /*  (186 0) */
    scBezFactor( 0x29b8,     0.1629710 ),   /*  (186 1) */
    scBezFactor( 0x6edb,     0.4330373 ),   /*  (186 2) */
    scBezFactor( 0x6230,     0.3835473 )    /*  (186 3) */
},
#endif
```

```
    scBezFactor( 0x71c1,     0.4443626 ),   /*  (172 2) */
    scBezFactor( 0x4da4,     0.3032951 )    /*  (172 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x08b9,     0.0340812 ),   /*  (173 0) */
    scBezFactor( 0x368e,     0.2131099 ),   /*  (173 1) */
    scBezFactor( 0x71b6,     0.4441929 ),   /*  (173 2) */
    scBezFactor( 0x4f01,     0.3086160 )    /*  (173 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x0869,     0.0328641 ),   /*  (174 0) */
    scBezFactor( 0x358e,     0.2092080 ),   /*  (174 1) */
    scBezFactor( 0x71a5,     0.4439292 ),   /*  (174 2) */
    scBezFactor( 0x5062,     0.3139987 )    /*  (174 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x081b,     0.0316764 ),   /*  (175 0) */
    scBezFactor( 0x348f,     0.2053097 ),   /*  (175 1) */
    scBezFactor( 0x718d,     0.4435703 ),   /*  (175 2) */
    scBezFactor( 0x51c7,     0.3194436 )    /*  (175 3) */
},
#endif


#ifdef SubDiv16
{
    scBezFactor( 0x07d0,     0.0305176 ),   /*  (176 0) */
    scBezFactor( 0x3390,     0.2014160 ),   /*  (176 1) */
    scBezFactor( 0x7170,     0.4431152 ),   /*  (176 2) */
    scBezFactor( 0x5330,     0.3249512 )    /*  (176 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x0785,     0.0293874 ),   /*  (177 0) */
    scBezFactor( 0x3291,     0.1975281 ),   /*  (177 1) */
    scBezFactor( 0x714b,     0.4425629 ),   /*  (177 2) */
    scBezFactor( 0x549d,     0.3305216 )    /*  (177 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x073d,     0.0282855 ),   /*  (178 0) */
    scBezFactor( 0x3192,     0.1936469 ),   /*  (178 1) */
    scBezFactor( 0x7121,     0.4419122 ),   /*  (178 2) */
    scBezFactor( 0x560e,     0.3361554 )    /*  (178 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x06f7,     0.0272115 ),   /*  (179 0) */
    scBezFactor( 0x3095,     0.1897736 ),   /*  (179 1) */
    scBezFactor( 0x70ef,     0.4411620 ),   /*  (179 2) */
    scBezFactor( 0x5783,     0.3418528 )    /*  (179 3) */
},
{
    scBezFactor( 0x0b7f,     0.0449163 ),   /*  (165 0) */
    scBezFactor( 0x3e8c,     0.2443251 ),   /*  (165 1) */
    scBezFactor( 0x7168,     0.4430071 ),   /*  (165 2) */
    scBezFactor( 0x448b,     0.2677515 )    /*  (165 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x0b1f,     0.0434518 ),   /*  (166 0) */
    scBezFactor( 0x3d8d,     0.2404332 ),   /*  (166 1) */
    scBezFactor( 0x7186,     0.4434657 ),   /*  (166 2) */
    scBezFactor( 0x45cc,     0.2726493 )    /*  (166 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x0ac1,     0.0420194 ),   /*  (167 0) */
    scBezFactor( 0x3c8d,     0.2365363 ),   /*  (167 1) */
    scBezFactor( 0x719f,     0.4438378 ),   /*  (167 2) */
    scBezFactor( 0x4711,     0.2776064 )    /*  (167 3) */
},
#endif


#ifdef SubDiv32
{
    scBezFactor( 0x0a66,     0.0406189 ),   /*  (168 0) */
    scBezFactor( 0x3b8e,     0.2326355 ),   /*  (168 1) */
    scBezFactor( 0x71b2,     0.4441223 ),   /*  (168 2) */
    scBezFactor( 0x485a,     0.2826233 )    /*  (168 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x0a0c,     0.0392498 ),   /*  (169 0) */
    scBezFactor( 0x3a8e,     0.2287318 ),   /*  (169 1) */
    scBezFactor( 0x71be,     0.4443181 ),   /*  (169 2) */
    scBezFactor( 0x49a6,     0.2877002 )    /*  (169 3) */
},
#endif

#ifdef SubDiv128
{
    scBezFactor( 0x09b4,     0.0379119 ),   /*  (170 0) */
    scBezFactor( 0x398e,     0.2248263 ),   /*  (170 1) */
    scBezFactor( 0x71c5,     0.4444242 ),   /*  (170 2) */
    scBezFactor( 0x4af7,     0.2928376 )    /*  (170 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x095e,     0.0366047 ),   /*  (171 0) */
    scBezFactor( 0x388e,     0.2209201 ),   /*  (171 1) */
    scBezFactor( 0x71c6,     0.4444394 ),   /*  (171 2) */
    scBezFactor( 0x4c4c,     0.2980358 )    /*  (171 3) */
},
#endif


#ifdef SubDiv64
{
    scBezFactor( 0x090b,     0.0353279 ),   /*  (172 0) */
    scBezFactor( 0x378e,     0.2170143 ),   /*  (172 1) */
```

```
#ifdef SubDiv128
{
    scBezFactor( 0x0e5c,     0.0560994 ),   /*  (158 0) */
    scBezFactor( 0x4576,     0.2713380 ),   /*  (158 1) */
    scBezFactor( 0x6ffd,     0.4374633 ),   /*  (158 2) */
    scBezFactor( 0x3c2f,     0.2350993 )    /*  (158 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x0ded,     0.0543995 ),   /*  (159 0) */
    scBezFactor( 0x447b,     0.2675112 ),   /*  (159 1) */
    scBezFactor( 0x7041,     0.4384977 ),   /*  (159 2) */
    scBezFactor( 0x3d55,     0.2395915 )    /*  (159 3) */
},
#endif


#ifdef SubDiv8
{
    scBezFactor( 0x0d80,     0.0527344 ),   /*  (160 0) */
    scBezFactor( 0x4380,     0.2636719 ),   /*  (160 1) */
    scBezFactor( 0x7080,     0.4394531 ),   /*  (160 2) */
    scBezFactor( 0x3e80,     0.2441406 )    /*  (160 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x0d15,     0.0511035 ),   /*  (161 0) */
    scBezFactor( 0x4283,     0.2598211 ),   /*  (161 1) */
    scBezFactor( 0x70b9,     0.4403284 ),   /*  (161 2) */
    scBezFactor( 0x3fad,     0.2487469 )    /*  (161 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x0cac,     0.0495067 ),   /*  (162 0) */
    scBezFactor( 0x4186,     0.2559600 ),   /*  (162 1) */
    scBezFactor( 0x70ed,     0.4411225 ),   /*  (162 2) */
    scBezFactor( 0x40df,     0.2534108 )    /*  (162 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x0c46,     0.0479434 ),   /*  (163 0) */
    scBezFactor( 0x4088,     0.2520896 ),   /*  (163 1) */
    scBezFactor( 0x711c,     0.4418344 ),   /*  (163 2) */
    scBezFactor( 0x4214,     0.2581326 )    /*  (163 3) */
},
#endif


#ifdef SubDiv64
{
    scBezFactor( 0x0be1,     0.0464134 ),   /*  (164 0) */
    scBezFactor( 0x3f8a,     0.2482109 ),   /*  (164 1) */
    scBezFactor( 0x7145,     0.4424629 ),   /*  (164 2) */
    scBezFactor( 0x434e,     0.2629128 )    /*  (164 3) */
},
#endif


#ifdef SubDiv256
```

```
    scBezFactor( 0x337f,     0.2011657 )    /*  (150 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x11a9,     0.0689998 ),   /*  (151 0) */
    scBezFactor( 0x4c35,     0.2976850 ),   /*  (151 1) */
    scBezFactor( 0x6d97,     0.4280993 ),   /*  (151 2) */
    scBezFactor( 0x3489,     0.2052159 )    /*  (151 3) */
},
#endif


#ifdef SubDiv32
{
    scBezFactor( 0x112a,     0.0670471 ),   /*  (152 0) */
    scBezFactor( 0x4b42,     0.2939758 ),   /*  (152 1) */
    scBezFactor( 0x6dfe,     0.4296570 ),   /*  (152 2) */
    scBezFactor( 0x3596,     0.2093201 )    /*  (152 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x10ac,     0.0651316 ),   /*  (153 0) */
    scBezFactor( 0x4a4d,     0.2902467 ),   /*  (153 1) */
    scBezFactor( 0x6e5f,     0.4311431 ),   /*  (153 2) */
    scBezFactor( 0x36a6,     0.2134786 )    /*  (153 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x1031,     0.0632529 ),   /*  (154 0) */
    scBezFactor( 0x4957,     0.2864985 ),   /*  (154 1) */
    scBezFactor( 0x6ebc,     0.4325566 ),   /*  (154 2) */
    scBezFactor( 0x37ba,     0.2176919 )    /*  (154 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x0fb8,     0.0614107 ),   /*  (155 0) */
    scBezFactor( 0x4861,     0.2827325 ),   /*  (155 1) */
    scBezFactor( 0x6f13,     0.4338965 ),   /*  (155 2) */
    scBezFactor( 0x38d2,     0.2219602 )    /*  (155 3) */
},
#endif


#ifdef SubDiv64
{
    scBezFactor( 0x0f42,     0.0596046 ),   /*  (156 0) */
    scBezFactor( 0x4769,     0.2789497 ),   /*  (156 1) */
    scBezFactor( 0x6f66,     0.4351616 ),   /*  (156 2) */
    scBezFactor( 0x39ed,     0.2262840 )    /*  (156 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x0ece,     0.0578343 ),   /*  (157 0) */
    scBezFactor( 0x4670,     0.2751512 ),   /*  (157 1) */
    scBezFactor( 0x6fb4,     0.4363509 ),   /*  (157 2) */
    scBezFactor( 0x3b0c,     0.2306636 )    /*  (157 3) */
},
#endif
```

```
    scBezFactor( 0x1604,     0.0860034 ),   /*  (143 0) */
    scBezFactor( 0x5396,     0.3265083 ),   /*  (143 1) */
    scBezFactor( 0x69c6,     0.4131920 ),   /*  (143 2) */
    scBezFactor( 0x2c9e,     0.1742963 )    /*  (143 3) */
},
#endif


#ifdef SubDiv16
{
    scBezFactor( 0x1570,     0.0837402 ),   /*  (144 0) */
    scBezFactor( 0x52b0,     0.3229980 ),   /*  (144 1) */
    scBezFactor( 0x6a50,     0.4152832 ),   /*  (144 2) */
    scBezFactor( 0x2d90,     0.1779785 )    /*  (144 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x14de,     0.0815172 ),   /*  (145 0) */
    scBezFactor( 0x51c8,     0.3194591 ),   /*  (145 1) */
    scBezFactor( 0x6ad4,     0.4173115 ),   /*  (145 2) */
    scBezFactor( 0x2e84,     0.1817122 )    /*  (145 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x144f,     0.0793338 ),   /*  (146 0) */
    scBezFactor( 0x50de,     0.3158927 ),   /*  (146 1) */
    scBezFactor( 0x6b55,     0.4192758 ),   /*  (146 2) */
    scBezFactor( 0x2f7c,     0.1854978 )    /*  (146 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x13c2,     0.0771897 ),   /*  (147 0) */
    scBezFactor( 0x4ff2,     0.3122998 ),   /*  (147 1) */
    scBezFactor( 0x6bd2,     0.4211749 ),   /*  (147 2) */
    scBezFactor( 0x3078,     0.1893355 )    /*  (147 3) */
},
#endif


#ifdef SubDiv64
{
    scBezFactor( 0x1338,     0.0750847 ),   /*  (148 0) */
    scBezFactor( 0x4f05,     0.3086815 ),   /*  (148 1) */
    scBezFactor( 0x6c4a,     0.4230080 ),   /*  (148 2) */
    scBezFactor( 0x3177,     0.1932259 )    /*  (148 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x12b1,     0.0730183 ),   /*  (149 0) */
    scBezFactor( 0x4e17,     0.3050389 ),   /*  (149 1) */
    scBezFactor( 0x6cbd,     0.4247738 ),   /*  (149 2) */
    scBezFactor( 0x3279,     0.1971691 )    /*  (149 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x122c,     0.0709901 ),   /*  (150 0) */
    scBezFactor( 0x4d26,     0.3013730 ),   /*  (150 1) */
    scBezFactor( 0x6d2d,     0.4264712 ),   /*  (150 2) */
```

```
#ifdef SubDiv32
{
    scBezFactor( 0x1a5e,     0.1029968 ),   /*  (136 0) */
    scBezFactor( 0x59a6,     0.3501892 ),   /*  (136 1) */
    scBezFactor( 0x659a,     0.3968811 ),   /*  (136 2) */
    scBezFactor( 0x2662,     0.1499329 )    /*  (136 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x19b6,     0.1004433 ),   /*  (137 0) */
    scBezFactor( 0x58cf,     0.3469092 ),   /*  (137 1) */
    scBezFactor( 0x663d,     0.3993829 ),   /*  (137 2) */
    scBezFactor( 0x273c,     0.1532646 )    /*  (137 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x1912,     0.0979323 ),   /*  (138 0) */
    scBezFactor( 0x57f5,     0.3435931 ),   /*  (138 1) */
    scBezFactor( 0x66de,     0.4018292 ),   /*  (138 2) */
    scBezFactor( 0x2819,     0.1566453 )    /*  (138 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x1870,     0.0954636 ),   /*  (139 0) */
    scBezFactor( 0x571a,     0.3402420 ),   /*  (139 1) */
    scBezFactor( 0x677a,     0.4042191 ),   /*  (139 2) */
    scBezFactor( 0x28fa,     0.1600754 )    /*  (139 3) */
},
#endif


#ifdef SubDiv64
{
    scBezFactor( 0x17d1,     0.0930367 ),   /*  (140 0) */
    scBezFactor( 0x563c,     0.3368568 ),   /*  (140 1) */
    scBezFactor( 0x6813,     0.4065514 ),   /*  (140 2) */
    scBezFactor( 0x29de,     0.1635551 )    /*  (140 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x1734,     0.0906512 ),   /*  (141 0) */
    scBezFactor( 0x555c,     0.3334388 ),   /*  (141 1) */
    scBezFactor( 0x68a8,     0.4088250 ),   /*  (141 2) */
    scBezFactor( 0x2ac6,     0.1670850 )    /*  (141 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x169b,     0.0883069 ),   /*  (142 0) */
    scBezFactor( 0x547a,     0.3299890 ),   /*  (142 1) */
    scBezFactor( 0x6939,     0.4110389 ),   /*  (142 2) */
    scBezFactor( 0x2bb0,     0.1706653 )    /*  (142 3) */
},
#endif


#ifdef SubDiv256
{
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x1f41,     0.1220931 ),   /*  (129 0) */
    scBezFactor( 0x5f3e,     0.3720476 ),   /*  (129 1) */
    scBezFactor( 0x60be,     0.3779066 ),   /*  (129 2) */
    scBezFactor( 0x20c1,     0.1279526 )    /*  (129 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x1e85,     0.1192317 ),   /*  (130 0) */
    scBezFactor( 0x5e7a,     0.3690505 ),   /*  (130 1) */
    scBezFactor( 0x6179,     0.3807664 ),   /*  (130 2) */
    scBezFactor( 0x2186,     0.1309514 )    /*  (130 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x1dcd,     0.1164153 ),   /*  (131 0) */
    scBezFactor( 0x5db2,     0.3660098 ),   /*  (131 1) */
    scBezFactor( 0x6232,     0.3835782 ),   /*  (131 2) */
    scBezFactor( 0x224d,     0.1339967 )    /*  (131 3) */
},
#endif


#ifdef SubDiv64
{
    scBezFactor( 0x1d17,     0.1136436 ),   /*  (132 0) */
    scBezFactor( 0x5ce8,     0.3629265 ),   /*  (132 1) */
    scBezFactor( 0x62e7,     0.3863411 ),   /*  (132 2) */
    scBezFactor( 0x2318,     0.1370888 )    /*  (132 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x1c65,     0.1109163 ),   /*  (133 0) */
    scBezFactor( 0x5c1b,     0.3598017 ),   /*  (133 1) */
    scBezFactor( 0x6399,     0.3890539 ),   /*  (133 2) */
    scBezFactor( 0x23e5,     0.1402281 )    /*  (133 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x1bb5,     0.1082330 ),   /*  (134 0) */
    scBezFactor( 0x5b4c,     0.3566365 ),   /*  (134 1) */
    scBezFactor( 0x6447,     0.3917155 ),   /*  (134 2) */
    scBezFactor( 0x24b6,     0.1434150 )    /*  (134 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x1b08,     0.1055933 ),   /*  (135 0) */
    scBezFactor( 0x5a7a,     0.3534320 ),   /*  (135 1) */
    scBezFactor( 0x64f2,     0.3943250 ),   /*  (135 2) */
    scBezFactor( 0x258a,     0.1466498 )    /*  (135 3) */
},
#endif
```

```
    scBezFactor( 0x64f2,     0.3943250 ),   /*  (121 1) */
    scBezFactor( 0x5a7a,     0.3534320 ),   /*  (121 2) */
    scBezFactor( 0x1b08,     0.1055933 )    /*  (121 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x24b6,     0.1434150 ),   /*  (122 0) */
    scBezFactor( 0x6447,     0.3917155 ),   /*  (122 1) */
    scBezFactor( 0x5b4c,     0.3566365 ),   /*  (122 2) */
    scBezFactor( 0x1bb5,     0.1082330 )    /*  (122 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x23e5,     0.1402281 ),   /*  (123 0) */
    scBezFactor( 0x6399,     0.3890539 ),   /*  (123 1) */
    scBezFactor( 0x5c1b,     0.3598017 ),   /*  (123 2) */
    scBezFactor( 0x1c65,     0.1109163 )    /*  (123 3) */
},
#endif


#ifdef SubDiv64
{
    scBezFactor( 0x2318,     0.1370888 ),   /*  (124 0) */
    scBezFactor( 0x62e7,     0.3863411 ),   /*  (124 1) */
    scBezFactor( 0x5ce8,     0.3629265 ),   /*  (124 2) */
    scBezFactor( 0x1d17,     0.1136436 )    /*  (124 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x224d,     0.1339967 ),   /*  (125 0) */
    scBezFactor( 0x6232,     0.3835782 ),   /*  (125 1) */
    scBezFactor( 0x5db2,     0.3660098 ),   /*  (125 2) */
    scBezFactor( 0x1dcd,     0.1164153 )    /*  (125 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x2186,     0.1309514 ),   /*  (126 0) */
    scBezFactor( 0x6179,     0.3807664 ),   /*  (126 1) */
    scBezFactor( 0x5e7a,     0.3690505 ),   /*  (126 2) */
    scBezFactor( 0x1e85,     0.1192317 )    /*  (126 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x20c1,     0.1279526 ),   /*  (127 0) */
    scBezFactor( 0x60be,     0.3779066 ),   /*  (127 1) */
    scBezFactor( 0x5f3e,     0.3720476 ),   /*  (127 2) */
    scBezFactor( 0x1f41,     0.1220931 )    /*  (127 3) */
},
#endif


#ifdef SubDiv2
{
    scBezFactor( 0x2000,     0.1250000 ),   /*  (128 0) */
    scBezFactor( 0x6000,     0.3750000 ),   /*  (128 1) */
    scBezFactor( 0x6000,     0.3750000 ),   /*  (128 2) */
    scBezFactor( 0x2000,     0.1250000 )    /*  (128 3) */
```

```
#ifdef SubDiv128
{
    scBezFactor( 0x2bb0,     0.1706653 ),   /*  (114 0) */
    scBezFactor( 0x6939,     0.4110389 ),   /*  (114 1) */
    scBezFactor( 0x547a,     0.3299890 ),   /*  (114 2) */
    scBezFactor( 0x169b,     0.0883069 )    /*  (114 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x2ac6,     0.1670850 ),   /*  (115 0) */
    scBezFactor( 0x68a8,     0.4088250 ),   /*  (115 1) */
    scBezFactor( 0x555c,     0.3334388 ),   /*  (115 2) */
    scBezFactor( 0x1734,     0.0906512 )    /*  (115 3) */
},
#endif


#ifdef SubDiv64
{
    scBezFactor( 0x29de,     0.1635551 ),   /*  (116 0) */
    scBezFactor( 0x6813,     0.4065514 ),   /*  (116 1) */
    scBezFactor( 0x563c,     0.3368568 ),   /*  (116 2) */
    scBezFactor( 0x17d1,     0.0930367 )    /*  (116 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x28fa,     0.1600754 ),   /*  (117 0) */
    scBezFactor( 0x677a,     0.4042191 ),   /*  (117 1) */
    scBezFactor( 0x571a,     0.3402420 ),   /*  (117 2) */
    scBezFactor( 0x1870,     0.0954636 )    /*  (117 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x2819,     0.1566453 ),   /*  (118 0) */
    scBezFactor( 0x66de,     0.4018292 ),   /*  (118 1) */
    scBezFactor( 0x57f5,     0.3435931 ),   /*  (118 2) */
    scBezFactor( 0x1912,     0.0979323 )    /*  (118 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x273c,     0.1532646 ),   /*  (119 0) */
    scBezFactor( 0x663d,     0.3993829 ),   /*  (119 1) */
    scBezFactor( 0x58cf,     0.3469092 ),   /*  (119 2) */
    scBezFactor( 0x19b6,     0.1004433 )    /*  (119 3) */
},
#endif


#ifdef SubDiv32
{
    scBezFactor( 0x2662,     0.1499329 ),   /*  (120 0) */
    scBezFactor( 0x659a,     0.3968811 ),   /*  (120 1) */
    scBezFactor( 0x59a6,     0.3501892 ),   /*  (120 2) */
    scBezFactor( 0x1a5e,     0.1029968 )    /*  (120 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x258a,     0.1466498 ),   /*  (121 0) */
```

```
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x3279,     0.1971691 ),   /*  (107 0) */
    scBezFactor( 0x6cbd,     0.4247738 ),   /*  (107 1) */
    scBezFactor( 0x4e17,     0.3050389 ),   /*  (107 2) */
    scBezFactor( 0x12b1,     0.0730183 )    /*  (107 3) */
},
#endif


#ifdef SubDiv64
{
    scBezFactor( 0x3177,     0.1932259 ),   /*  (108 0) */
    scBezFactor( 0x6c4a,     0.4230080 ),   /*  (108 1) */
    scBezFactor( 0x4f05,     0.3086815 ),   /*  (108 2) */
    scBezFactor( 0x1338,     0.0750847 )    /*  (108 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x3078,     0.1893355 ),   /*  (109 0) */
    scBezFactor( 0x6bd2,     0.4211749 ),   /*  (109 1) */
    scBezFactor( 0x4ff2,     0.3122998 ),   /*  (109 2) */
    scBezFactor( 0x13c2,     0.0771897 )    /*  (109 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x2f7c,     0.1854978 ),   /*  (110 0) */
    scBezFactor( 0x6b55,     0.4192758 ),   /*  (110 1) */
    scBezFactor( 0x50de,     0.3158927 ),   /*  (110 2) */
    scBezFactor( 0x144f,     0.0793338 )    /*  (110 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x2e84,     0.1817122 ),   /*  (111 0) */
    scBezFactor( 0x6ad4,     0.4173115 ),   /*  (111 1) */
    scBezFactor( 0x51c8,     0.3194591 ),   /*  (111 2) */
    scBezFactor( 0x14de,     0.0815172 )    /*  (111 3) */
},
#endif


#ifdef SubDiv16
{
    scBezFactor( 0x2d90,     0.1779785 ),   /*  (112 0) */
    scBezFactor( 0x6a50,     0.4152832 ),   /*  (112 1) */
    scBezFactor( 0x52b0,     0.3229980 ),   /*  (112 2) */
    scBezFactor( 0x1570,     0.0837402 )    /*  (112 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x2c9e,     0.1742963 ),   /*  (113 0) */
    scBezFactor( 0x69c6,     0.4131920 ),   /*  (113 1) */
    scBezFactor( 0x5396,     0.3265083 ),   /*  (113 2) */
    scBezFactor( 0x1604,     0.0860034 )    /*  (113 3) */
},
#endif
```

```
    scBezFactor( 0x4670,     0.2751512 ),   /*  (99 2) */
    scBezFactor( 0x0ece,     0.0578343 )    /*  (99 3) */
},
#endif


#ifdef SubDiv64
{
    scBezFactor( 0x39ed,     0.2262840 ),   /*  (100 0) */
    scBezFactor( 0x6f66,     0.4351616 ),   /*  (100 1) */
    scBezFactor( 0x4769,     0.2789497 ),   /*  (100 2) */
    scBezFactor( 0x0f42,     0.0596046 )    /*  (100 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x38d2,     0.2219602 ),   /*  (101 0) */
    scBezFactor( 0x6f13,     0.4338965 ),   /*  (101 1) */
    scBezFactor( 0x4861,     0.2827325 ),   /*  (101 2) */
    scBezFactor( 0x0fb8,     0.0614107 )    /*  (101 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x37ba,     0.2176919 ),   /*  (102 0) */
    scBezFactor( 0x6ebc,     0.4325566 ),   /*  (102 1) */
    scBezFactor( 0x4957,     0.2864985 ),   /*  (102 2) */
    scBezFactor( 0x1031,     0.0632529 )    /*  (102 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x36a6,     0.2134786 ),   /*  (103 0) */
    scBezFactor( 0x6e5f,     0.4311431 ),   /*  (103 1) */
    scBezFactor( 0x4a4d,     0.2902467 ),   /*  (103 2) */
    scBezFactor( 0x10ac,     0.0651316 )    /*  (103 3) */
},
#endif


#ifdef SubDiv32
{
    scBezFactor( 0x3596,     0.2093201 ),   /*  (104 0) */
    scBezFactor( 0x6dfe,     0.4296570 ),   /*  (104 1) */
    scBezFactor( 0x4b42,     0.2939758 ),   /*  (104 2) */
    scBezFactor( 0x112a,     0.0670471 )    /*  (104 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x3489,     0.2052159 ),   /*  (105 0) */
    scBezFactor( 0x6d97,     0.4280993 ),   /*  (105 1) */
    scBezFactor( 0x4c35,     0.2976850 ),   /*  (105 2) */
    scBezFactor( 0x11a9,     0.0689998 )    /*  (105 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x337f,     0.2011657 ),   /*  (106 0) */
    scBezFactor( 0x6d2d,     0.4264712 ),   /*  (106 1) */
    scBezFactor( 0x4d26,     0.3013730 ),   /*  (106 2) */
    scBezFactor( 0x122c,     0.0709901 )    /*  (106 3) */
},
{
    scBezFactor( 0x434e,     0.2629128 ),   /*  (92 0) */
    scBezFactor( 0x7145,     0.4424629 ),   /*  (92 1) */
    scBezFactor( 0x3f8a,     0.2482109 ),   /*  (92 2) */
    scBezFactor( 0x0be1,     0.0464134 )    /*  (92 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x4214,     0.2581326 ),   /*  (93 0) */
    scBezFactor( 0x711c,     0.4418344 ),   /*  (93 1) */
    scBezFactor( 0x4088,     0.2520896 ),   /*  (93 2) */
    scBezFactor( 0x0c46,     0.0479434 )    /*  (93 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x40df,     0.2534108 ),   /*  (94 0) */
    scBezFactor( 0x70ed,     0.4411225 ),   /*  (94 1) */
    scBezFactor( 0x4186,     0.2559600 ),   /*  (94 2) */
    scBezFactor( 0x0cac,     0.0495067 )    /*  (94 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x3fad,     0.2487469 ),   /*  (95 0) */
    scBezFactor( 0x70b9,     0.4403284 ),   /*  (95 1) */
    scBezFactor( 0x4283,     0.2598211 ),   /*  (95 2) */
    scBezFactor( 0x0d15,     0.0511035 )    /*  (95 3) */
},
#endif


#ifdef SubDiv8
{
    scBezFactor( 0x3e80,     0.2441406 ),   /*  (96 0) */
    scBezFactor( 0x7080,     0.4394531 ),   /*  (96 1) */
    scBezFactor( 0x4380,     0.2636719 ),   /*  (96 2) */
    scBezFactor( 0x0d80,     0.0527344 )    /*  (96 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x3d55,     0.2395915 ),   /*  (97 0) */
    scBezFactor( 0x7041,     0.4384977 ),   /*  (97 1) */
    scBezFactor( 0x447b,     0.2675112 ),   /*  (97 2) */
    scBezFactor( 0x0ded,     0.0543995 )    /*  (97 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x3c2f,     0.2350993 ),   /*  (98 0) */
    scBezFactor( 0x6ffd,     0.4374633 ),   /*  (98 1) */
    scBezFactor( 0x4576,     0.2713380 ),   /*  (98 2) */
    scBezFactor( 0x0e5c,     0.0560994 )    /*  (98 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x3b0c,     0.2306636 ),   /*  (99 0) */
    scBezFactor( 0x6fb4,     0.4363509 ),   /*  (99 1) */
```

```
#ifdef SubDiv256
{
    scBezFactor( 0x4c4c,     0.2980358 ),   /*  (85 0) */
    scBezFactor( 0x71c6,     0.4444394 ),   /*  (85 1) */
    scBezFactor( 0x388e,     0.2209201 ),   /*  (85 2) */
    scBezFactor( 0x095e,     0.0366047 )    /*  (85 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x4af7,     0.2928376 ),   /*  (86 0) */
    scBezFactor( 0x71c5,     0.4444242 ),   /*  (86 1) */
    scBezFactor( 0x398e,     0.2248263 ),   /*  (86 2) */
    scBezFactor( 0x09b4,     0.0379119 )    /*  (86 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x49a6,     0.2877002 ),   /*  (87 0) */
    scBezFactor( 0x71be,     0.4443181 ),   /*  (87 1) */
    scBezFactor( 0x3a8e,     0.2287318 ),   /*  (87 2) */
    scBezFactor( 0x0a0c,     0.0392498 )    /*  (87 3) */
},
#endif


#ifdef SubDiv32
{
    scBezFactor( 0x485a,     0.2826233 ),   /*  (88 0) */
    scBezFactor( 0x71b2,     0.4441223 ),   /*  (88 1) */
    scBezFactor( 0x3b8e,     0.2326355 ),   /*  (88 2) */
    scBezFactor( 0x0a66,     0.0406189 )    /*  (88 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x4711,     0.2776064 ),   /*  (89 0) */
    scBezFactor( 0x719f,     0.4438378 ),   /*  (89 1) */
    scBezFactor( 0x3c8d,     0.2365363 ),   /*  (89 2) */
    scBezFactor( 0x0ac1,     0.0420194 )    /*  (89 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x45cc,     0.2726493 ),   /*  (90 0) */
    scBezFactor( 0x7186,     0.4434657 ),   /*  (90 1) */
    scBezFactor( 0x3d8d,     0.2404332 ),   /*  (90 2) */
    scBezFactor( 0x0b1f,     0.0434518 )    /*  (90 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x448b,     0.2677515 ),   /*  (91 0) */
    scBezFactor( 0x7168,     0.4430071 ),   /*  (91 1) */
    scBezFactor( 0x3e8c,     0.2443251 ),   /*  (91 2) */
    scBezFactor( 0x0b7f,     0.0449163 )    /*  (91 3) */
},
#endif


#ifdef SubDiv64
```

```
    scBezFactor( 0x06f7,     0.0272115 )    /*  (77 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x560e,     0.3361554 ),   /*  (78 0) */
    scBezFactor( 0x7121,     0.4419122 ),   /*  (78 1) */
    scBezFactor( 0x3192,     0.1936469 ),   /*  (78 2) */
    scBezFactor( 0x073d,     0.0282855 )    /*  (78 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x549d,     0.3305216 ),   /*  (79 0) */
    scBezFactor( 0x714b,     0.4425629 ),   /*  (79 1) */
    scBezFactor( 0x3291,     0.1975281 ),   /*  (79 2) */
    scBezFactor( 0x0785,     0.0293874 )    /*  (79 3) */
},
#endif


#ifdef SubDiv16
{
    scBezFactor( 0x5330,     0.3249512 ),   /*  (80 0) */
    scBezFactor( 0x7170,     0.4431152 ),   /*  (80 1) */
    scBezFactor( 0x3390,     0.2014160 ),   /*  (80 2) */
    scBezFactor( 0x07d0,     0.0305176 )    /*  (80 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x51c7,     0.3194436 ),   /*  (81 0) */
    scBezFactor( 0x718d,     0.4435703 ),   /*  (81 1) */
    scBezFactor( 0x348f,     0.2053097 ),   /*  (81 2) */
    scBezFactor( 0x081b,     0.0316764 )    /*  (81 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x5062,     0.3139987 ),   /*  (82 0) */
    scBezFactor( 0x71a5,     0.4439292 ),   /*  (82 1) */
    scBezFactor( 0x358e,     0.2092080 ),   /*  (82 2) */
    scBezFactor( 0x0869,     0.0328641 )    /*  (82 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x4f01,     0.3086160 ),   /*  (83 0) */
    scBezFactor( 0x71b6,     0.4441929 ),   /*  (83 1) */
    scBezFactor( 0x368e,     0.2131099 ),   /*  (83 2) */
    scBezFactor( 0x08b9,     0.0340812 )    /*  (83 3) */
},
#endif


#ifdef SubDiv64
{
    scBezFactor( 0x4da4,     0.3032951 ),   /*  (84 0) */
    scBezFactor( 0x71c1,     0.4443626 ),   /*  (84 1) */
    scBezFactor( 0x378e,     0.2170143 ),   /*  (84 2) */
    scBezFactor( 0x090b,     0.0353279 )    /*  (84 3) */
},
#endif
```

```
    scBezFactor( 0x6230,     0.3835473 ),   /*  (70 0) */
    scBezFactor( 0x6edb,     0.4330373 ),   /*  (70 1) */
    scBezFactor( 0x29b8,     0.1629710 ),   /*  (70 2) */
    scBezFactor( 0x053b,     0.0204444 )    /*  (70 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x609c,     0.3773943 ),   /*  (71 0) */
    scBezFactor( 0x6f3c,     0.4345134 ),   /*  (71 1) */
    scBezFactor( 0x2ab0,     0.1667592 ),   /*  (71 2) */
    scBezFactor( 0x0576,     0.0213332 )    /*  (71 3) */
},
#endif


#ifdef SubDiv32
{
    scBezFactor( 0x5f0e,     0.3713074 ),   /*  (72 0) */
    scBezFactor( 0x6f96,     0.4358826 ),   /*  (72 1) */
    scBezFactor( 0x2baa,     0.1705627 ),   /*  (72 2) */
    scBezFactor( 0x05b2,     0.0222473 )    /*  (72 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x5d83,     0.3652863 ),   /*  (73 0) */
    scBezFactor( 0x6fe8,     0.4371459 ),   /*  (73 1) */
    scBezFactor( 0x2ca4,     0.1743806 ),   /*  (73 2) */
    scBezFactor( 0x05ef,     0.0231872 )    /*  (73 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x5bfd,     0.3593307 ),   /*  (74 0) */
    scBezFactor( 0x7034,     0.4383044 ),   /*  (74 1) */
    scBezFactor( 0x2d9f,     0.1782117 ),   /*  (74 2) */
    scBezFactor( 0x062e,     0.0241532 )    /*  (74 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x5a7b,     0.3534401 ),   /*  (75 0) */
    scBezFactor( 0x7079,     0.4393592 ),   /*  (75 1) */
    scBezFactor( 0x2e9b,     0.1820549 ),   /*  (75 2) */
    scBezFactor( 0x066f,     0.0251457 )    /*  (75 3) */
},
#endif


#ifdef SubDiv64
{
    scBezFactor( 0x58fd,     0.3476143 ),   /*  (76 0) */
    scBezFactor( 0x70b8,     0.4403114 ),   /*  (76 1) */
    scBezFactor( 0x2f97,     0.1859093 ),   /*  (76 2) */
    scBezFactor( 0x06b2,     0.0261650 )    /*  (76 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x5783,     0.3418528 ),   /*  (77 0) */
    scBezFactor( 0x70ef,     0.4411620 ),   /*  (77 1) */
    scBezFactor( 0x3095,     0.1897736 ),   /*  (77 2) */
```

```
#ifdef SubDiv256
{
    scBezFactor( 0x6db2,     0.4285012 ),   /*  (63 0) */
    scBezFactor( 0x6b6c,     0.4196203 ),   /*  (63 1) */
    scBezFactor( 0x2310,     0.1369745 ),   /*  (63 2) */
    scBezFactor( 0x03d0,     0.0149040 )    /*  (63 3) */
},
#endif


#ifdef SubDiv4
{
    scBezFactor( 0x6c00,     0.4218750 ),   /*  (64 0) */
    scBezFactor( 0x6c00,     0.4218750 ),   /*  (64 1) */
    scBezFactor( 0x2400,     0.1406250 ),   /*  (64 2) */
    scBezFactor( 0x0400,     0.0156250 )    /*  (64 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x6a52,     0.4153175 ),   /*  (65 0) */
    scBezFactor( 0x6c8c,     0.4240152 ),   /*  (65 1) */
    scBezFactor( 0x24f0,     0.1442984 ),   /*  (65 2) */
    scBezFactor( 0x0430,     0.0163689 )    /*  (65 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x68a8,     0.4088283 ),   /*  (66 0) */
    scBezFactor( 0x6d11,     0.4260421 ),   /*  (66 1) */
    scBezFactor( 0x25e2,     0.1479936 ),   /*  (66 2) */
    scBezFactor( 0x0463,     0.0171361 )    /*  (66 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x6704,     0.4024070 ),   /*  (67 0) */
    scBezFactor( 0x6d8e,     0.4279566 ),   /*  (67 1) */
    scBezFactor( 0x26d6,     0.1517095 ),   /*  (67 2) */
    scBezFactor( 0x0496,     0.0179269 )    /*  (67 3) */
},
#endif


#ifdef SubDiv64
{
    scBezFactor( 0x6563,     0.3960533 ),   /*  (68 0) */
    scBezFactor( 0x6e04,     0.4297600 ),   /*  (68 1) */
    scBezFactor( 0x27cb,     0.1554451 ),   /*  (68 2) */
    scBezFactor( 0x04cc,     0.0187416 )    /*  (68 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x63c7,     0.3897669 ),   /*  (69 0) */
    scBezFactor( 0x6e73,     0.4314532 ),   /*  (69 1) */
    scBezFactor( 0x28c1,     0.1591993 ),   /*  (69 2) */
    scBezFactor( 0x0503,     0.0195807 )    /*  (69 3) */
},
#endif


#ifdef SubDiv128
{
},
#endif


#ifdef SubDiv32
{
    scBezFactor( 0x7a12,     0.4768372 ),   /*  (56 0) */
    scBezFactor( 0x668a,     0.4005432 ),   /*  (56 1) */
    scBezFactor( 0x1cb6,     0.1121521 ),   /*  (56 2) */
    scBezFactor( 0x02ae,     0.0104675 )    /*  (56 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x783f,     0.4697203 ),   /*  (57 0) */
    scBezFactor( 0x6754,     0.4036290 ),   /*  (57 1) */
    scBezFactor( 0x1d98,     0.1156123 ),   /*  (57 2) */
    scBezFactor( 0x02d3,     0.0110384 )    /*  (57 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x7671,     0.4626746 ),   /*  (58 0) */
    scBezFactor( 0x6816,     0.4065928 ),   /*  (58 1) */
    scBezFactor( 0x1e7d,     0.1191030 ),   /*  (58 2) */
    scBezFactor( 0x02fa,     0.0116296 )    /*  (58 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x74a8,     0.4556997 ),   /*  (59 0) */
    scBezFactor( 0x68d0,     0.4094358 ),   /*  (59 1) */
    scBezFactor( 0x1f64,     0.1226229 ),   /*  (59 2) */
    scBezFactor( 0x0322,     0.0122415 )    /*  (59 3) */
},
#endif


#ifdef SubDiv64
{
    scBezFactor( 0x72e4,     0.4487953 ),   /*  (60 0) */
    scBezFactor( 0x6983,     0.4121590 ),   /*  (60 1) */
    scBezFactor( 0x204c,     0.1261711 ),   /*  (60 2) */
    scBezFactor( 0x034b,     0.0128746 )    /*  (60 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x7124,     0.4419610 ),   /*  (61 0) */
    scBezFactor( 0x6a2d,     0.4147634 ),   /*  (61 1) */
    scBezFactor( 0x2137,     0.1297465 ),   /*  (61 2) */
    scBezFactor( 0x0376,     0.0135291 )    /*  (61 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x6f69,     0.4351964 ),   /*  (62 0) */
    scBezFactor( 0x6ad0,     0.4172502 ),   /*  (62 1) */
    scBezFactor( 0x2223,     0.1333480 ),   /*  (62 2) */
    scBezFactor( 0x03a2,     0.0142055 )    /*  (62 3) */
},
#endif
```

```
    scBezFactor( 0x5f10,     0.3713379 ),   /*  (48 1) */
    scBezFactor( 0x15f0,     0.0856934 ),   /*  (48 2) */
    scBezFactor( 0x01b0,     0.0065918 )    /*  (48 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x8757,     0.5286779 ),   /*  (49 0) */
    scBezFactor( 0x601c,     0.3754379 ),   /*  (49 1) */
    scBezFactor( 0x16c0,     0.0888718 ),   /*  (49 2) */
    scBezFactor( 0x01cb,     0.0070124 )    /*  (49 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x8563,     0.5210528 ),   /*  (50 0) */
    scBezFactor( 0x6120,     0.3794074 ),   /*  (50 1) */
    scBezFactor( 0x1793,     0.0920892 ),   /*  (50 2) */
    scBezFactor( 0x01e8,     0.0074506 )    /*  (50 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x8374,     0.5135015 ),   /*  (51 0) */
    scBezFactor( 0x621c,     0.3832474 ),   /*  (51 1) */
    scBezFactor( 0x1868,     0.0953445 ),   /*  (51 2) */
    scBezFactor( 0x0206,     0.0079066 )    /*  (51 3) */
},
#endif


#ifdef SubDiv64
{
    scBezFactor( 0x818a,     0.5060234 ),   /*  (52 0) */
    scBezFactor( 0x630f,     0.3869591 ),   /*  (52 1) */
    scBezFactor( 0x1940,     0.0986366 ),   /*  (52 2) */
    scBezFactor( 0x0225,     0.0083809 )    /*  (52 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x7fa5,     0.4986183 ),   /*  (53 0) */
    scBezFactor( 0x63fa,     0.3905434 ),   /*  (53 1) */
    scBezFactor( 0x1a1a,     0.1019645 ),   /*  (53 2) */
    scBezFactor( 0x0245,     0.0088738 )    /*  (53 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x7dc4,     0.4912858 ),   /*  (54 0) */
    scBezFactor( 0x64dd,     0.3940015 ),   /*  (54 1) */
    scBezFactor( 0x1af6,     0.1053271 ),   /*  (54 2) */
    scBezFactor( 0x0267,     0.0093856 )    /*  (54 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x7be9,     0.4840255 ),   /*  (55 0) */
    scBezFactor( 0x65b7,     0.3973344 ),   /*  (55 1) */
    scBezFactor( 0x1bd5,     0.1087233 ),   /*  (55 2) */
    scBezFactor( 0x0289,     0.0099167 )    /*  (55 3) */
```

```
#ifdef SubDiv256
{
    scBezFactor( 0x97a5,     0.5923733 ),   /*  (41 0) */
    scBezFactor( 0x56c1,     0.3388926 ),   /*  (41 1) */
    scBezFactor( 0x108b,     0.0646260 ),   /*  (41 2) */
    scBezFactor( 0x010d,     0.0041080 )    /*  (41 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x958a,     0.5841460 ),   /*  (42 0) */
    scBezFactor( 0x580c,     0.3439364 ),   /*  (42 1) */
    scBezFactor( 0x1147,     0.0675015 ),   /*  (42 2) */
    scBezFactor( 0x0121,     0.0044160 )    /*  (42 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x9374,     0.5759953 ),   /*  (43 0) */
    scBezFactor( 0x594d,     0.3488422 ),   /*  (43 1) */
    scBezFactor( 0x1207,     0.0704235 ),   /*  (43 2) */
    scBezFactor( 0x0136,     0.0047390 )    /*  (43 3) */
},
#endif


#ifdef SubDiv64
{
    scBezFactor( 0x9163,     0.5679207 ),   /*  (44 0) */
    scBezFactor( 0x5a86,     0.3536110 ),   /*  (44 1) */
    scBezFactor( 0x12c9,     0.0733910 ),   /*  (44 2) */
    scBezFactor( 0x014c,     0.0050774 )    /*  (44 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x8f57,     0.5599219 ),   /*  (45 0) */
    scBezFactor( 0x5bb5,     0.3582439 ),   /*  (45 1) */
    scBezFactor( 0x138f,     0.0764027 ),   /*  (45 2) */
    scBezFactor( 0x0163,     0.0054315 )    /*  (45 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x8d4f,     0.5519986 ),   /*  (46 0) */
    scBezFactor( 0x5cdc,     0.3627419 ),   /*  (46 1) */
    scBezFactor( 0x1457,     0.0794578 ),   /*  (46 2) */
    scBezFactor( 0x017c,     0.0058017 )    /*  (46 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x8b4d,     0.5441504 ),   /*  (47 0) */
    scBezFactor( 0x5dfa,     0.3671063 ),   /*  (47 1) */
    scBezFactor( 0x1522,     0.0825550 ),   /*  (47 2) */
    scBezFactor( 0x0195,     0.0061883 )    /*  (47 3) */
},
#endif


#ifdef SubDiv16
{
    scBezFactor( 0x8950,     0.5363770 ),   /*  (48 0) */
```

```
#endif


#ifdef SubDiv128
{
    scBezFactor( 0xa6f2,     0.6521373 ),   /*  (34 0) */
    scBezFactor( 0x4cb4,     0.2996306 ),   /*  (34 1) */
    scBezFactor( 0x0bbf,     0.0458894 ),   /*  (34 2) */
    scBezFactor( 0x0099,     0.0023427 )    /*  (34 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0xa4b3,     0.6433643 ),   /*  (35 0) */
    scBezFactor( 0x4e40,     0.3056708 ),   /*  (35 1) */
    scBezFactor( 0x0c64,     0.0484094 ),   /*  (35 2) */
    scBezFactor( 0x00a7,     0.0025555 )    /*  (35 3) */
},
#endif


#ifdef SubDiv64
{
    scBezFactor( 0xa279,     0.6346703 ),   /*  (36 0) */
    scBezFactor( 0x4fc2,     0.3115654 ),   /*  (36 1) */
    scBezFactor( 0x0d0d,     0.0509834 ),   /*  (36 2) */
    scBezFactor( 0x00b6,     0.0027809 )    /*  (36 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0xa045,     0.6260549 ),   /*  (37 0) */
    scBezFactor( 0x513b,     0.3173155 ),   /*  (37 1) */
    scBezFactor( 0x0db9,     0.0536104 ),   /*  (37 2) */
    scBezFactor( 0x00c5,     0.0030192 )    /*  (37 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0x9e15,     0.6175179 ),   /*  (38 0) */
    scBezFactor( 0x52ab,     0.3229222 ),   /*  (38 1) */
    scBezFactor( 0x0e68,     0.0562892 ),   /*  (38 2) */
    scBezFactor( 0x00d6,     0.0032706 )    /*  (38 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0x9beb,     0.6090589 ),   /*  (39 0) */
    scBezFactor( 0x5411,     0.3283866 ),   /*  (39 1) */
    scBezFactor( 0x0f1b,     0.0590188 ),   /*  (39 2) */
    scBezFactor( 0x00e7,     0.0035357 )    /*  (39 3) */
},
#endif


#ifdef SubDiv32
{
    scBezFactor( 0x99c6,     0.6006775 ),   /*  (40 0) */
    scBezFactor( 0x556e,     0.3337097 ),   /*  (40 1) */
    scBezFactor( 0x0fd2,     0.0617981 ),   /*  (40 2) */
    scBezFactor( 0x00fa,     0.0038147 )    /*  (40 3) */
},
#endif
```

```
    scBezFactor( 0x071e,     0.0278020 ),   /*  (26 2) */
    scBezFactor( 0x0044,     0.0010476 )    /*  (26 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0xb73e,     0.7157915 ),   /*  (27 0) */
    scBezFactor( 0x40d0,     0.2531839 ),   /*  (27 1) */
    scBezFactor( 0x07a4,     0.0298514 ),   /*  (27 2) */
    scBezFactor( 0x004c,     0.0011732 )    /*  (27 3) */
},
#endif


#ifdef SubDiv64
{
    scBezFactor( 0xb4da,     0.7064552 ),   /*  (28 0) */
    scBezFactor( 0x42a1,     0.2602730 ),   /*  (28 1) */
    scBezFactor( 0x082e,     0.0319633 ),   /*  (28 2) */
    scBezFactor( 0x0055,     0.0013084 )    /*  (28 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0xb27b,     0.6972005 ),   /*  (29 0) */
    scBezFactor( 0x4467,     0.2672090 ),   /*  (29 1) */
    scBezFactor( 0x08bd,     0.0341368 ),   /*  (29 2) */
    scBezFactor( 0x005f,     0.0014537 )    /*  (29 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0xb022,     0.6880269 ),   /*  (30 0) */
    scBezFactor( 0x4624,     0.2739930 ),   /*  (30 1) */
    scBezFactor( 0x094f,     0.0363708 ),   /*  (30 2) */
    scBezFactor( 0x0069,     0.0016093 )    /*  (30 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0xadce,     0.6789342 ),   /*  (31 0) */
    scBezFactor( 0x47d7,     0.2806261 ),   /*  (31 1) */
    scBezFactor( 0x09e5,     0.0386640 ),   /*  (31 2) */
    scBezFactor( 0x0074,     0.0017757 )    /*  (31 3) */
},
#endif


#ifdef SubDiv8
{
    scBezFactor( 0xab80,     0.6699219 ),   /*  (32 0) */
    scBezFactor( 0x4980,     0.2871094 ),   /*  (32 1) */
    scBezFactor( 0x0a80,     0.0410156 ),   /*  (32 2) */
    scBezFactor( 0x0080,     0.0019531 )    /*  (32 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0xa936,     0.6609897 ),   /*  (33 0) */
    scBezFactor( 0x4b1f,     0.2934439 ),   /*  (33 1) */
    scBezFactor( 0x0b1d,     0.0434244 ),   /*  (33 2) */
    scBezFactor( 0x008c,     0.0021420 )    /*  (33 3) */
},
{
    scBezFactor( 0xcb20,     0.7934602 ),   /*  (19 0) */
    scBezFactor( 0x30da,     0.1908322 ),   /*  (19 1) */
    scBezFactor( 0x03ea,     0.0152988 ),   /*  (19 2) */
    scBezFactor( 0x001a,     0.0004088 )    /*  (19 3) */
},
#endif


#ifdef SubDiv64
{
    scBezFactor( 0xc890,     0.7834587 ),   /*  (20 0) */
    scBezFactor( 0x32fd,     0.1991844 ),   /*  (20 1) */
    scBezFactor( 0x0452,     0.0168800 ),   /*  (20 2) */
    scBezFactor( 0x001f,     0.0004768 )    /*  (20 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0xc606,     0.7735416 ),   /*  (21 0) */
    scBezFactor( 0x3516,     0.2073750 ),   /*  (21 1) */
    scBezFactor( 0x04be,     0.0185314 ),   /*  (21 2) */
    scBezFactor( 0x0024,     0.0005520 )    /*  (21 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0xc382,     0.7637086 ),   /*  (22 0) */
    scBezFactor( 0x3724,     0.2154050 ),   /*  (22 1) */
    scBezFactor( 0x052f,     0.0202518 ),   /*  (22 2) */
    scBezFactor( 0x0029,     0.0006347 )    /*  (22 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0xc103,     0.7539592 ),   /*  (23 0) */
    scBezFactor( 0x3928,     0.2232755 ),   /*  (23 1) */
    scBezFactor( 0x05a4,     0.0220401 ),   /*  (23 2) */
    scBezFactor( 0x002f,     0.0007252 )    /*  (23 3) */
},
#endif


#ifdef SubDiv32
{
    scBezFactor( 0xbe8a,     0.7442932 ),   /*  (24 0) */
    scBezFactor( 0x3b22,     0.2309875 ),   /*  (24 1) */
    scBezFactor( 0x061e,     0.0238953 ),   /*  (24 2) */
    scBezFactor( 0x0036,     0.0008240 )    /*  (24 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0xbc15,     0.7347102 ),   /*  (25 0) */
    scBezFactor( 0x3d11,     0.2385423 ),   /*  (25 1) */
    scBezFactor( 0x069b,     0.0258163 ),   /*  (25 2) */
    scBezFactor( 0x003d,     0.0009313 )    /*  (25 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0xb9a7,     0.7252097 ),   /*  (26 0) */
    scBezFactor( 0x3ef5,     0.2459407 ),   /*  (26 1) */
```

```
#ifdef SubDiv64
{
    scBezFactor( 0xdda9,     0.8658638 ),   /*  (12 0) */
    scBezFactor( 0x20b4,     0.1277504 ),   /*  (12 1) */
    scBezFactor( 0x019b,     0.0062828 ),   /*  (12 2) */
    scBezFactor( 0x0006,     0.0001030 )    /*  (12 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0xdaf2,     0.8552615 ),   /*  (13 0) */
    scBezFactor( 0x2323,     0.1372642 ),   /*  (13 1) */
    scBezFactor( 0x01e1,     0.0073434 ),   /*  (13 2) */
    scBezFactor( 0x0008,     0.0001310 )    /*  (13 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0xd841,     0.8447461 ),   /*  (14 0) */
    scBezFactor( 0x2588,     0.1466088 ),   /*  (14 1) */
    scBezFactor( 0x022b,     0.0084815 ),   /*  (14 2) */
    scBezFactor( 0x000a,     0.0001636 )    /*  (14 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0xd595,     0.8343173 ),   /*  (15 0) */
    scBezFactor( 0x27e1,     0.1557854 ),   /*  (15 1) */
    scBezFactor( 0x027b,     0.0096962 ),   /*  (15 2) */
    scBezFactor( 0x000d,     0.0002012 )    /*  (15 3) */
},
#endif


#ifdef SubDiv16
{
    scBezFactor( 0xd2f0,     0.8239746 ),   /*  (16 0) */
    scBezFactor( 0x2a30,     0.1647949 ),   /*  (16 1) */
    scBezFactor( 0x02d0,     0.0109863 ),   /*  (16 2) */
    scBezFactor( 0x0010,     0.0002441 )    /*  (16 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0xd04f,     0.8137178 ),   /*  (17 0) */
    scBezFactor( 0x2c73,     0.1736385 ),   /*  (17 1) */
    scBezFactor( 0x0329,     0.0123509 ),   /*  (17 2) */
    scBezFactor( 0x0013,     0.0002928 )    /*  (17 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0xcdb5,     0.8035464 ),   /*  (18 0) */
    scBezFactor( 0x2eac,     0.1823173 ),   /*  (18 1) */
    scBezFactor( 0x0387,     0.0137887 ),   /*  (18 2) */
    scBezFactor( 0x0016,     0.0003476 )    /*  (18 3) */
},
#endif


#ifdef SubDiv256
```

```
    scBezFactor( 0x0000,     0.0000038 )    /*  (4 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0xf14a,     0.9425432 ),   /*  (5 0) */
    scBezFactor( 0x0e6b,     0.0563273 ),   /*  (5 1) */
    scBezFactor( 0x0049,     0.0011221 ),   /*  (5 2) */
    scBezFactor( 0x0000,     0.0000075 )    /*  (5 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0xee6b,     0.9313226 ),   /*  (6 0) */
    scBezFactor( 0x112a,     0.0670552 ),   /*  (6 1) */
    scBezFactor( 0x0069,     0.0016093 ),   /*  (6 2) */
    scBezFactor( 0x0000,     0.0000129 )    /*  (6 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0xeb91,     0.9201913 ),   /*  (7 0) */
    scBezFactor( 0x13de,     0.0776065 ),   /*  (7 1) */
    scBezFactor( 0x008e,     0.0021817 ),   /*  (7 2) */
    scBezFactor( 0x0001,     0.0000204 )    /*  (7 3) */
},
#endif


#ifdef SubDiv32
{
    scBezFactor( 0xe8be,     0.9091492 ),   /*  (8 0) */
    scBezFactor( 0x1686,     0.0879822 ),   /*  (8 1) */
    scBezFactor( 0x00ba,     0.0028381 ),   /*  (8 2) */
    scBezFactor( 0x0002,     0.0000305 )    /*  (8 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0xe5f0,     0.8981957 ),   /*  (9 0) */
    scBezFactor( 0x1922,     0.0981833 ),   /*  (9 1) */
    scBezFactor( 0x00ea,     0.0035775 ),   /*  (9 2) */
    scBezFactor( 0x0002,     0.0000435 )    /*  (9 3) */
},
#endif


#ifdef SubDiv128
{
    scBezFactor( 0xe328,     0.8873305 ),   /*  (10 0) */
    scBezFactor( 0x1bb3,     0.1082110 ),   /*  (10 1) */
    scBezFactor( 0x0120,     0.0043988 ),   /*  (10 2) */
    scBezFactor( 0x0003,     0.0000596 )    /*  (10 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0xe065,     0.8765534 ),   /*  (11 0) */
    scBezFactor( 0x1e39,     0.1180664 ),   /*  (11 1) */
    scBezFactor( 0x015b,     0.0053009 ),   /*  (11 2) */
    scBezFactor( 0x0005,     0.0000793 )    /*  (11 3) */
},
#endif
```

```
/****************************************************************************

    File:       SCBEZBLE.C


    $Header: /Projects/Toolbox/ct/SCBEZBLE.CPP 2     5/30/97 8:45a Wmanis $

    Contains:   the blending values for computing beziers

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

****************************************************************************/

/* this contains standard function values, we use tables instead of actually
 * computing the value
 */

#include "scbezier.h"

scBezBlendValue bezblend[scBezBlendSize] = {

/* this one appears in all */
{
    scBezFactor( 0x0000,     1.0000000 ),   /*  (0 0) */
    scBezFactor( 0x0000,     0.0000000 ),   /*  (0 1) */
    scBezFactor( 0x0000,     0.0000000 ),   /*  (0 2) */
    scBezFactor( 0x0000,     0.0000000 )    /*  (0 3) */
},

#ifdef SubDiv256
{
    scBezFactor( 0xfd02,     0.9883270 ),   /*  (1 0) */
    scBezFactor( 0x02fa,     0.0116274 ),   /*  (1 1) */
    scBezFactor( 0x0002,     0.0000456 ),   /*  (1 2) */
    scBezFactor( 0x0000,     0.0000001 )    /*  (1 3) */
},
#endif

#ifdef SubDiv128
{
    scBezFactor( 0xfa0b,     0.9767451 ),   /*  (2 0) */
    scBezFactor( 0x05e8,     0.0230727 ),   /*  (2 1) */
    scBezFactor( 0x000b,     0.0001817 ),   /*  (2 2) */
    scBezFactor( 0x0000,     0.0000005 )    /*  (2 3) */
},
#endif


#ifdef SubDiv256
{
    scBezFactor( 0xf71a,     0.9652541 ),   /*  (3 0) */
    scBezFactor( 0x08ca,     0.0343371 ),   /*  (3 1) */
    scBezFactor( 0x001a,     0.0004072 ),   /*  (3 2) */
    scBezFactor( 0x0000,     0.0000016 )    /*  (3 3) */
},
#endif


#ifdef SubDiv64
{
    scBezFactor( 0xf42f,     0.9538536 ),   /*  (4 0) */
    scBezFactor( 0x0ba0,     0.0454216 ),   /*  (4 1) */
    scBezFactor( 0x002f,     0.0007210 ),   /*  (4 2) */
```

```
class scBinarySortedArray : public scSizeableArray<T> {
public:
    int     Find( const T&, int* insertIndex = 0 ) const;
    int     Find1( const T&, const CT&, int* insertIndex = 0 ) const;
    int     SortInsert( const T& );
};

/* ==================================================================== */

template <class T, class CT>
class scBinarySortedArrayD : public scSizeableArrayD<T> {
public:
    int     Find( const T&, int* insertIndex = 0 ) const;
    int     Find1( const T&, const CT&, int* insertIndex = 0 ) const;
    int     SortInsert( const T& );
};

/* ==================================================================== */

#ifdef DEFINE_TEMPLATES
#include "scarray.cpp"
#endif

#endif
```

```
    void        GrowSlots( int );
    void        ClearMem( unsigned );

    unsigned    elemSlots_;         // num of elements potentially in allocated space
    unsigned    blockSize_ : 16;        // for growing and shrinking we grow in greater
                                    // than one element unit - this is that unit
                                    // typically 4
    unsigned    retainMem_ : 1;     // do not shrink memory if this is set
};


/* ================================================================== */

template <class T>
class scSizeableArrayD : public scArray<T> {
public:
                scSizeableArrayD();
                ~scSizeableArrayD();

    void        Remove( int );
    void        RemoveAll( void );

    T&          Grow();

    int         Append( const T& );
    void        Insert( int, const T& );

    void        Set( int, const T& );

    void        SetNumSlots( int numSlots );

    void        SetRetainMem( int tf )
                    {
                        retainMem_ = tf ? 1 : 0;
                    }
    int         GetRetainMem( void ) const
                    {
                        return retainMem_;
                    }

    void        exch( const scSizeableArrayD<T>& );

private:
    void        constructItem( int );
    void        deleteItem( int );

    void        MoreSlots( void )
                    {
                        GrowSlots( blockSize_ );
                    }
    void        ShrinkSlots( void )
                    {
                        SetNumSlots( numItems_ );
                    }

    void        SizeSlots( unsigned );
    void        GrowSlots( int );
    void        ClearMem( unsigned );

    unsigned    elemSlots_;         // num of elements potentially in allocated space
    unsigned    blockSize_ : 16;    // for growing and shrinking we grow in greater
                                    // than one element unit - this is that unit
                                    // typically 4
    unsigned    retainMem_ : 1;     // do not shrink memory if this is set
};


/* ================================================================== */

// CT is a comparison class - it must have a method
// Compare( const T&, const T& )

template <class T, class CT>
```

```
{
    scArray<T>  array = (scArray<T>&)arr;

    int t = array.numItems_;
    array.numItems_ = numItems_;
    numItems_ = array.numItems_;

    T* tmp = array.items_;
    array.items_ = items_;
    items_ = tmp;
}
#endif

/* ==================================================================== */

template <class T>
class scStaticArray : public scArray<T> {
public:
                scStaticArray( int num, T* mem = 0 );
                ~scStaticArray();
};

/* ==================================================================== */

template <class T>
class scStaticArrayD : public scArray<T> {
public:
                scStaticArrayD( int num, T* mem = 0 );
                ~scStaticArrayD();
};

/* ==================================================================== */

template <class T>
class scSizeableArray : public scArray<T> {
public:
                scSizeableArray();
                ~scSizeableArray();

    void        Remove( int );
    void        RemoveAll( void );
    int         Append( const T& );

    T&          Grow();

    void        Insert( int, const T& );

    void        Set( int, const T& );

    void        SetNumSlots( int numSlots );

    void        SetRetainMem( int tf )
                    {
                        retainMem_ = tf ? 1 : 0;
                    }
    int         GetRetainMem( void ) const
                    {
                        return retainMem_;
                    }

//  void        exch( const scSizeableArray<T>& );

private:
    void        MoreSlots( void )
                    {
                        GrowSlots( blockSize_ );
                    }
    void        ShrinkSlots( void )
                    {
                        SetNumSlots( numItems_ );
                    }

    void        SizeSlots( unsigned );
```

```
                scArray( int num, T* mem ) :
                        numItems_( num ),
                        items_( mem ) {}



    int         numItems_;
    T*          items_;

#ifndef __MWERKS__
                    // the following are not declared because of the
                    // deep vs shallow copy problem
                scArray( const scArray& );      // not declared
    scArray&    operator=( const scArray& );    // not declared
#endif
};


template <class T>
#ifndef __MWERKS__
inline
#endif
int  scArray<T>::NumItems() const
    {
        return numItems_;
    }

template <class T>
#ifndef __MWERKS__
inline
#endif
T&  scArray<T>::operator[]( int n )
    {
        return items_[ n ];
    }

template <class T>
#ifndef __MWERKS__
inline
#endif
const T&  scArray<T>::operator[]( int n ) const
    {
        return items_[ n ];
    }


template <class T>
#ifndef __MWERKS__
inline
#endif
T*  scArray<T>::ptr()
    {
        return items_;
    }


template <class T>
#ifndef __MWERKS__
inline
#endif
const T*  scArray<T>::ptr() const
    {
        return items_;
    }


#if 0
template <class T>
#ifndef __MWERKS__
inline
#endif
void scArray<T>::exch( const scArray<T>& arr )
```

```
/****************************************************************************

    File:       SCARRAY.H


    $Header: /btoolbox/lib/SCARRAY.H 3     3/31/96 3:48p Will $

    Contains:   Templates for Vector

    Written by: Manis

    Copyright (c) 1989-95 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

****************************************************************************/


#ifndef _H_SCARRAY
#define _H_SCARRAY

#ifdef _DEBUG
    #define ifdebug( x )    x
#else
    #define ifdebug( x )
#endif

/* ===================================================================== */

template <class T>
class scAutoDelete {
public:
        scAutoDelete( T* ptr = 0 ): ptr_( ptr ) {}
        ~scAutoDelete()     { freePtr(); }

  T*    deref() const       { return ptr_; }
  T*    operator->() const  { return ptr_; }
  T&    operator*() const   { return *ptr_; }

  void  operator=( T* p )   { if ( ptr_ ) freePtr(); ptr_ = p; }
  int   operator==( const scAutoDelete<T>& p )
                            { return ptr_ == p.deref(); }

private:
  void  freePtr()           { delete ptr_, ptr_ = 0; }
  T*    ptr_;
};

/* ===================================================================== */

template <class T>
class scArray {
public:
    int         NumItems( void ) const;

    T&          operator[]( int n );
    const T&    operator[]( int n ) const;

    T*          ptr( void );
    const T*    ptr( void ) const;

//  void exch( const scArray<T>& );
protected:

                scArray() :
                    numItems_(0),
                    items_(0){}
        int found = ct.Compare1( val, items_[index] );
        if ( found == 0 )
            return index;
        else if ( found < 0 )
            high = index - 1, insertIndex = index;
        else
            low = index + 1, insertIndex = index + 1;
    }

    insertIndexP ? *insertIndexP = insertIndex : 0;
    return -1;
}

template<class T, class CT>
int scBinarySortedArrayD<T,CT>::SortInsert( const T& item )
{
    int     index;
    if ( Find( item, &index ) < 0 )
        Insert( index, item );
    return index;
}


#endif /* scDEFINE_TEMPLATES */
```

[illegible]

```
    if ( items_ == 0 )
        throw( -1 );

    items_ = (T*)realloc( items_, sizeof(T) * ( elemSlots_ + newItems ) );
    elemSlots_ += newItems;
    ClearMem( oldSize );
}


template <class T>
void scSizeableArrayD<T>::ClearMem( unsigned oldsize )
{
        // either we do need to clear memory or we have shrunk it
    if ( oldsize >= elemSlots_ )
        return;

    for ( unsigned index = oldsize; index < elemSlots_; index++ )
        constructItem( index );
}

#if 0
template <class T>
void scSizeableArrayD<T>::exch( const scSizeableArrayD<T>& arr )
{
    scArray<T>::operator=( arr );

    scSizeableArrayD<T>& array = (scSizeableArrayD<T>&)arr;

    unsigned tmp = elemSlots_;
    elemSlots_ = array.elemSlots_;
    array.elemSlots_ = tmp;

    tmp = blockSize_;
    blockSize_ = array.blockSize_;
    array.blockSize_ = blockSize_;

    tmp = retainMem_;
    retainMem_ = array.retainMem_;
    array.retainMem_ = retainMem_;
}
#endif

template<class T, class CT>
int scBinarySortedArrayD<T,CT>::Find( const T& val, int* insertIndexP ) const
{
    int     low      = 0;
    int     high     = numItems_ - 1;
    int insertIndex = 0;

    while ( low <= high ) {
        int index = (low + high) / 2;
        int found = CT::Compare( val, items_[index] );
        if ( found == 0 )
            return index;
        else if ( found < 0 )
            high = index - 1, insertIndex = index;
        else
            low = index + 1, insertIndex = index + 1;
    }

    insertIndexP ? *insertIndexP = insertIndex : 0;
    return -1;
}

template<class T, class CT>
int scBinarySortedArrayD<T,CT>::Find1( const T& val, const CT& ct, int* insertIndexP ) const
{
    int     low      = 0;
    int     high     = numItems_ - 1;
    int insertIndex = 0;

    while ( low <= high ) {
        int index = (low + high) / 2;
    SizeSlots( numBlocks * blockSize_ );
}

template <class T>
int scSizeableArrayD<T>::Append( const T& elem )
{
    SetNumSlots( numItems_ + 1 );
    items_[numItems_] = elem;
    return numItems_++;
}


template <class T>
T& scSizeableArrayD<T>::Grow()
{
    SetNumSlots( numItems_ + 1 );
    return items_[numItems_++];
}


template <class T>
void scSizeableArrayD<T>::Insert( int index, const T& elem )
{
    SetNumSlots( numItems_ + 1 );

    if ( numItems_ - index > 0 )
        memmove( items_ + index + 1, items_ + index,
                 ( numItems_ - index ) * sizeof(T) );

    constructItem( index );
    items_[index] = elem;
    numItems_++;
}

template <class T>
void scSizeableArrayD<T>::Set( int index, const T& elem )
{
    if ( index >= numItems_ ) {
        SetNumSlots( index + 1 );
        numItems_ = index + 1;
    }

    items_[index] = elem;
}


template <class T>
void scSizeableArrayD<T>::SizeSlots( unsigned numItems )
{
        // do not shrink if we are retaining memory or if no resizing is
        // necessary
        //
    if ( elemSlots_ == numItems || ( numItems < elemSlots_ && retainMem_ ) )
        return;

    stAssert( numItems >= blockSize_ );

    long oldSize = elemSlots_;

    if ( items_ == 0 )
        throw( -1 );

    items_ = (T*)realloc( items_, sizeof(T) * numItems );
    elemSlots_ = numItems;
    ClearMem( oldSize );
}


template <class T>
void scSizeableArrayD<T>::GrowSlots( int newItems )
{
    int oldSize = elemSlots_;
    }
    delete [] items_;
}

#define kInitBlockSize  4

template <class T>
scSizeableArrayD<T>::scSizeableArrayD() :
    elemSlots_( 4 ),
    blockSize_( kInitBlockSize ),
    scArray<T>( 0, (T*)malloc( kInitBlockSize * sizeof( T ) ) )
{
    ClearMem( 0 );
}

template <class T>
scSizeableArrayD<T>::~scSizeableArrayD()
{
    for ( unsigned i = 0; i < elemSlots_; i++)
        deleteItem( i );

    if ( items_ )
        free( items_ );
}

template <class T>
void scSizeableArrayD<T>::constructItem( int index )
{
    (void)new( items_ + index ) T;
}

template <class T>
void scSizeableArrayD<T>::deleteItem( int index )
{
#ifdef __WATCOMC__
    items_[index].~T();
#elif defined(_MAC)
    T& tp = items_[index];
    tp.~T();
#else
    items_[index].T::~T();
#endif
}

template <class T>
void scSizeableArrayD<T>::Remove( int index )
{
    stAssert( index < numItems_ );
    deleteItem( index );

    if ( numItems_ - index - 1 )
        memmove( items_ +index, items_ +index + 1,
                    ( numItems_ - index - 1 ) * sizeof( T )  );

    numItems_ -= 1;

    ShrinkSlots();
}

template <class T>
void scSizeableArrayD<T>::RemoveAll()
{
    for ( int i = 0; i < numItems_; i++)
        deleteItem( i );

    numItems_ = 0;
    ShrinkSlots();
}

template <class T>
void scSizeableArrayD<T>::SetNumSlots( int numSlots )
{
    int numBlocks = stMAX( 1, numSlots / blockSize_ + ( numSlots % blockSize_ ? 1 : 0 ) );
```

```
template<class T, class CT>
int scBinarySortedArray<T,CT>::Find( const T& val, int* insertIndexP ) const
{
    int     low      = 0;
    int     high     = numItems_ - 1;
    int insertIndex = 0;

    while ( low <= high ) {
        int index = (low + high) / 2;
        int found = CT::Compare( val, items_[index] );
        if ( found == 0 )
            return index;
        else if ( found < 0 )
            high = index - 1, insertIndex = index;
        else
            low = index + 1, insertIndex = index + 1;
    }

    insertIndexP ? *insertIndexP = insertIndex : 0;
    return -1;
}

template<class T, class CT>
int scBinarySortedArray<T,CT>::Find1( const T& val, const CT& ct, int* insertIndexP ) const
{
    int     low      = 0;
    int     high     = numItems_ - 1;
    int insertIndex = 0;

    while ( low <= high ) {
        int index = (low + high) / 2;
        int found = ct.Compare1( val, items_[index] );
        if ( found == 0 )
            return index;
        else if ( found < 0 )
            high = index - 1, insertIndex = index;
        else
            low = index + 1, insertIndex = index + 1;
    }

    insertIndexP ? *insertIndexP = insertIndex : 0;
    return -1;
}

template<class T, class CT>
int scBinarySortedArray<T,CT>::SortInsert( const T& item )
{
    int     index;
    if ( Find( item, &index ) < 0 )
        Insert( index, item );
    return index;
}

////

template <class T>
scStaticArrayD<T>::scStaticArrayD( int num, T* mem ) :
    scArray<T>( num, mem ? mem : new T[num] )
{
}

template <class T>
scStaticArrayD<T>::~scStaticArrayD()
{
    for ( int i = 0; i < numItems_; i++) {
#ifdef __WATCOMC__
        items_[i].~T();
#elif defined(_MAC)
        T& tp = items_[i];
        tp.~T();
#else
        items_[i].T::~T();
#endif
```

```
    items_[index] = elem;
}


template <class T>
void scSizeableArray<T>::SizeSlots( unsigned numItems )
{
        // do not shrink if we are retaining memory or if no resizing is
        // necessary
        //
    if ( elemSlots_ == numItems || ( numItems < elemSlots_ && retainMem_ ) )
        return;

    stAssert( numItems >= blockSize_ );

    long oldSize = elemSlots_;

    if ( items_ == 0 )
        throw( -1 );

    items_ = (T*)realloc( items_, sizeof(T) * numItems );
    elemSlots_ = numItems;
    ClearMem( oldSize );
}


template <class T>
void scSizeableArray<T>::GrowSlots( int newItems )
{
    int oldSize = elemSlots_;

    if ( items_ == 0 )
        throw( -1 );

    items_ = (T*)realloc( items_, sizeof(T) * ( elemSlots_ + newItems ) );
    elemSlots_ += newItems;
    ClearMem( oldSize );
}


template <class T>
void scSizeableArray<T>::ClearMem( unsigned oldsize )
{
        // either we do need to clear memory or we have shrunk it
    if ( oldsize >= elemSlots_ )
        return;

    for ( unsigned index = oldsize; index < elemSlots_; index++ )
        memset( items_ + index, 0, sizeof(T) );
}

#if 0
template <class T>
void scSizeableArray<T>::exch( const scSizeableArray<T>& arr )
{
    scArray<T>::operator=( arr );

    scSizeableArray<T>& array = (scSizeableArray<T>&)arr;

    unsigned tmp = elemSlots_;
    elemSlots_ = array.elemSlots_;
    array.elemSlots_ = tmp;

    tmp = blockSize_;
    blockSize_ = array.blockSize_;
    array.blockSize_ = blockSize_;

    tmp = retainMem_;
    retainMem_ = array.retainMem_;
    array.retainMem_ = retainMem_;
}
#endif
```

```
}

template <class T>
scSizeableArray<T>::~scSizeableArray()
{
    if ( items_ )
        free( items_ );
}

template <class T>
void scSizeableArray<T>::Remove( int index )
{
    stAssert( index < numItems_ );

    memmove( items_ +index, items_ +index + 1,
             ( numItems_ - index - 1 ) * sizeof( T )  );

    numItems_ -= 1;

    ShrinkSlots();
}

template <class T>
void scSizeableArray<T>::RemoveAll()
{
    numItems_ = 0;
    ShrinkSlots();
}

template <class T>
void scSizeableArray<T>::SetNumSlots( int numSlots )
{
    int numBlocks = stMAX( 1, numSlots / blockSize_ + ( numSlots % blockSize_ ? 1 : 0 ) );
    SizeSlots( numBlocks * blockSize_ );
}

template <class T>
int scSizeableArray<T>::Append( const T& elem )
{
    SetNumSlots( numItems_ + 1 );
    items_[numItems_] = elem;
    return numItems_++;
}

template <class T>
T& scSizeableArray<T>::Grow()
{
    SetNumSlots( numItems_ + 1 );
    return items_[numItems_++];
}


template <class T>
void scSizeableArray<T>::Insert( int index, const T& elem )
{
    SetNumSlots( numItems_ + 1 );

    if ( numItems_ - index > 0 )
        memmove( items_ + index + 1, items_ + index,
                 ( numItems_ - index ) * sizeof(T) );

    items_[index] = elem;
    numItems_++;
}

template <class T>
void scSizeableArray<T>::Set( int index, const T& elem )
{
    if ( index >= numItems_ ) {
        SetNumSlots( index + 1 );
        numItems_ = index + 1;
    }
```

```
/***************************************************************************

    File:      SCARRAY.CPP


    $Header: /btoolbox/lib/SCARRAY.CPP 3     3/31/96 3:48p Will $

    Contains:       Templates for Vector

    Written by:     Manis

    Copyright (c) 1989-95 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

***************************************************************************/


#ifdef DEFINE_TEMPLATES

#if defined( __MWERKS__ )
#include <stdlib.h>
#else
#include <malloc.h>
#endif

#include <string.h>
#include <assert.h>

inline
void *operator new(size_t, void *p)
{
    return p;
}

#ifndef scAssert
    #define stAssert    assert
#else
    #define stAssert    scAssert
#endif

#define stMAX( a, b )       ((a)>(b)?(a):(b))
#define stMIN( a, b )       ((a)<(b)?(a):(b))


template <class T>
scStaticArray<T>::scStaticArray( int num, T* mem ) :
    scArray<T>( num, mem ? mem : new T[num] )
{
}

template <class T>
scStaticArray<T>::~scStaticArray()
{
    delete [] items_;
}

#define kInitBlockSize  4

template <class T>
scSizeableArray<T>::scSizeableArray() :
    elemSlots_( 4 ),
    blockSize_( kInitBlockSize ),
    scArray<T>( 0, (T*)malloc( kInitBlockSize * sizeof( T ) ) )
{
    ClearMem( 0 );
```

```
//
typedef COLORREF                        APPColor;       /* color reference */

// @type APPDrwCtx | An abstract type/magic cookie that the Composition Toolbox
// may use to pass thru drawing contexts.
//
typedef CAGText*                        APPDrwCtx;      // drawing context

// @type APPFont | An abstract type/magic cookie that the Composition Toolbox
// may use to retrieve and specify font information.
//
typedef const scChar*                   APPFont;

// @type APPRender | An abstract type/magic cookie that the Composition Toolbox
// may use to specify font information plus additional drawing attributes
// that the client may wish to use (e.g. drop shadow ). Typically used when
// the traditional values returned by the font sub-system in Quickdraw or
// GDI would not suffice.
// @xref <t APPFont>
typedef struct RenderDef*               APPRender;

// @type TypeSpec | An abstract type/magic cookie that the Composition Toolbox
// may use to retrieve typographic state information.
//
#include "refcnt.h"
class stSpec : public RefCount
{
};
typedef class RefCountPtr<stSpec>       TypeSpec;
typedef class RefCountPtr<stSpec>       scFontRender;

// @type APPColumn | An abstract type/magic cookie to be filled in
// appropriately by the client.
//
typedef CAGText*                        APPColumn;

// @type APPCtxPtr | An abstract type/magic cookie for use in file i/o.
// @xref <t IOFuncPtr>
typedef CAGText*                        APPCtxPtr;

/* ==================================================================== */

#endif
```

```
/****************************************************************************

    File:       SCAPPTYP.H

    $Header: /Projects/Toolbox/ct/SCAPPTYP.H 2     5/30/97 8:45a Wmanis $

    Contains:   Defintion by client of data types.

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.


    Use this to define application types for proper type checking,
    these are types that are by and large passede thru the Compostion
    Toolboxt and thus type information is superfulous

@doc

 ****************************************************************************/

#ifndef _H_SCAPPTYP
#define _H_SCAPPTYP

#ifdef _WINDOWS
        #include <windows.h>
#endif

/* ==================================================================== */
/* ==================================================================== */

class DemoView;
class CAGText;
class ApplIOContext;

enum {
    Japanese    = 0,
    English,
    Spanish,
    Italian,
    Portuguese,
    French,
    German,
    Dutch,
    Bokmal,
    Nynorsk,
    Swedish,
    Danish,
    Icelandic,
    Greek,
    Turkish,
    Russian,
    Croatian,
    Finnish,
    Miscellaneous,
    MAX_LANGUAGES
};

// @type APPLanguage | An abstract type/magic cookie that the Composition Toolbox
// may use to specify hyphenation language.
typedef short                           APPLanguage;


// @type APPColor | An abstract type/magic cookie that the Composition Toolbox
// may use to specify color.
```

```
/***************************************************************************

    File:      SCAPPTEX.H


    $Header: /Projects/Toolbox/ct/SCAPPTEX.H 2     5/30/97 8:45a Wmanis $

    Contains:  The class for passing content plus typo state
               back between client and toolbox.

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

@doc
***************************************************************************/

#ifndef _H_SCAPPTEX
#define _H_SCAPPTEX

#ifdef SCMACINTOSH
#pragma once
#endif

#include "sctypes.h"

class stTextImportExport {
public:
    static  stTextImportExport& MakeTextImportExport( int encoding = 1 );

    virtual void    release() = 0;

        // writing
    virtual void    StartPara( TypeSpec& ) = 0;
    virtual void    SetParaSpec( TypeSpec& ) = 0;
    virtual void    PutString( const uchar*, int, TypeSpec& ) = 0;
    virtual void    PutString( stUnivString&, TypeSpec& ) = 0;
    virtual void    PutChar( UCS2, TypeSpec& ) = 0;

        // reading
    virtual int     NextPara( TypeSpec& ) = 0;
    virtual int     GetChar( UCS2&, TypeSpec& ) = 0;
    virtual void    reset() = 0;
    virtual void    resetpara() = 0;
};


#endif /* _H_SCAPPTEX */
```

```
}

/* ================================================================== */
```

```
    pindex_ = paras_.NumItems() - 1;
}

/* ================================================================= */

void stTIEImp::SetParaSpec( TypeSpec& ts )
{
    stPara& p = currentPara();
    p.setparaspec( ts );
}

/* ================================================================= */

void stTIEImp::PutString( const uchar* str, int len, TypeSpec& ts )
{
    stPara& p = currentPara();
    p.append( ts );
    p.append( str, len );
}

/* ================================================================= */

void stTIEImp::PutString( stUnivString& ustr, TypeSpec& ts )
{
    stPara& p = currentPara();
    p.append( ts );
    p.append( ustr );
}

/* ================================================================= */

void stTIEImp::PutChar( UCS2 ch, TypeSpec& ts )
{
    stPara& p = currentPara();
    p.append( ts );
    p.append( ch );
}

/* ================================================================= */
// reading

int stTIEImp::NextPara( TypeSpec& ts )
{
    pindex_++;
    if ( pindex_ < paras_.NumItems() ) {
        ts = paras_[pindex_].paraspec();
        paras_[pindex_].validate();
        return pindex_;
    }
    return -1;
}

/* ================================================================= */

int stTIEImp::GetChar( UCS2& ch, TypeSpec& ts )
{
    stPara& p = currentPara();
    return p.get( ch, ts );
}

/* ================================================================= */

void stTIEImp::reset()
{
    pindex_ = -1;
}

/* ================================================================= */

void stTIEImp::resetpara()
{
    stPara& p = currentPara();
    p.reset();
```

```
class stTIEImp : public stTextImportExport {
public:
                    stTIEImp();
                    ~stTIEImp();

    void            release();

        // writing
    virtual void    StartPara( TypeSpec& );
    virtual void    SetParaSpec( TypeSpec& );
    virtual void    PutString( const uchar*, int, TypeSpec& );
    virtual void    PutString( stUnivString&, TypeSpec& );
    virtual void    PutChar( UCS2, TypeSpec& );

        // reading
    virtual int     NextPara( TypeSpec& );
    virtual int     GetChar( UCS2&, TypeSpec& );
    virtual void    reset();
    virtual void    resetpara();

protected:
    stPara&         currentPara();

    int32                       pindex_;
    scSizeableArrayD<stPara>    paras_;
};


/* ==================================================================== */

stTextImportExport& stTextImportExport::MakeTextImportExport( int encoding )
{
    stTIEImp* stimp = new stTIEImp();
    return *stimp;
}

/* ==================================================================== */

stTIEImp::stTIEImp()
{
    reset();
}

/* ==================================================================== */

stTIEImp::~stTIEImp()
{
}

/* ==================================================================== */

void stTIEImp::release()
{
    delete this;
}

/* ==================================================================== */

stPara& stTIEImp::currentPara()
{
    return paras_[pindex_];
}

/* ==================================================================== */
// importing

void stTIEImp::StartPara( TypeSpec& ts )
{
    if ( paras_.NumItems() > 0 )
        paras_[pindex_].complete();

    stPara newPara( ts );
    paras_.Append( newPara );
```

```
void stPara::append( stUnivString& ustr )
{
    for ( unsigned i = 0; i < ustr.len; i++ )
        ch_.Append( (UCS2)ustr.ptr[i] );
}

/* ==================================================================== */

void stPara::append( UCS2 ch )
{
    ch_.Append( ch );
}

/* ==================================================================== */

stPara& stPara::operator=( const stPara&  p )
{
    ch_.RemoveAll();
    for ( int i = 0; i < p.ch_.NumItems(); i++ )
        ch_.Append( p.ch_[i] );

    choffset_ = p.choffset_;

    specs_.RemoveAll();
    for ( i = 0; i < p.specs_.NumItems(); i++ )
        specs_.Append( p.specs_[i] );

    paraspec_ = p.paraspec_;
    return *this;
}

/* ==================================================================== */

int stPara::get( UCS2& ch, TypeSpec& spec )
{
    if ( choffset_ < ch_.NumItems() ) {
        spec = specs_.SpecAtOffset( choffset_ );
        ch = ch_[choffset_++];
        return choffset_;
    }
    return 0;
}

/* ==================================================================== */

int stPara::validate() const
{
    scAssert( paraspec_.ptr() );
    specs_.DebugRun( "stPara::validate" );
    return 1;
}

/* ==================================================================== */

void stPara::setparaspec( TypeSpec& ts )
{
    paraspec_ = ts;
    scAssert( ch_.NumItems() == 0 );
    specs_.AppendSpec( ts, 0 );
}

/* ==================================================================== */

int stPara::complete()
{
    if ( specs_.NumItems() == 1 )
        specs_.AppendSpec( paraspec_, 0 );

    return validate();
}

/* ==================================================================== */
```

```
/*************************************************************************

    File:       SCAPPTEX.C


    $Header: /Projects/Toolbox/ct/SCAPPTEX.CPP 2     5/30/97 8:45a Wmanis $

    Contains:
        This module takes a handle to APPTextRun structure and manages the
        memory (specs & chars) associated with it. Locking and unlocking. It
        also maintains some internal structures that used for reading the text
        between the lock and unlock calls. Refer to the
        APPTextRun in SCTextExch.h.

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

**************************************************************************/

#include "scapptex.h"
#include "scparagr.h"

/* ================================================================== */

stPara::stPara( ) :
    paraspec_( 0 )
{
    reset();
}

/* ================================================================== */

stPara::stPara( TypeSpec& pspec ) :
    paraspec_( pspec )
{
    if ( pspec.ptr() )
        specs_.AppendSpec( pspec, 0 );
    reset();
}

/* ================================================================== */

stPara::~stPara()
{
}

/* ================================================================== */

void stPara::append( TypeSpec& ts )
{
    if ( ts != specs_.SpecAtOffset( ch_.NumItems() ) )
        specs_.AppendSpec( ts, ch_.NumItems() );
}

/* ================================================================== */

void stPara::append( const uchar* ch, int len )
{
    for ( int i = 0; i < len; i++ )
        ch_.Append( (UCS2)ch[i] );
}

/* ================================================================== */
```

```
void scIMPL_EXPORT      SCDebugParaSpecs( scSelection* );

void scIMPL_EXPORT      SCSTR_Debug( scStream* );

#endif /* DEBUG */


#endif /* _H_SCAPPINT */
//</pre></html>
```

```
    virtual void    range( scStreamLocation&, scStreamLocation& ) = 0;
};


status SCSTR_GetContUnitIter( scStream*, stContUnitIter*& );
status SCSEL_GetContUnitIter( scSelection*, stContUnitIter*& );


/*==============================================*/

class stFindIter {
public:
    virtual void    release() = 0;
    virtual void    reset() = 0;
    virtual int     setselection( scSelection* ) = 0;
    virtual int     next() = 0;
    virtual int     replacetoken( stUnivString& ) = 0;
    virtual void    range( scStreamLocation&, scStreamLocation& ) = 0;
};


status SCSTR_GetFindIter( scStream*,
                          stUnivString&,
                          const SearchState&,
                          stFindIter*& );
status SCSEL_GetFindIter( scSelection*,
                          stUnivString&,
                          const SearchState&,
                          stFindIter*& );

/*==============================================*/


#ifdef scRubiSupprt
class scAnnotation;

// take the select id and find the nth annotation within the selection,
// if no annotation is found - noAction is returned
// [ ] [ ]
status scIMPL_EXPORT    SCSEL_GetAnnotation( scSelection*,
                                             int nth,
                                             scAnnotation& );

// apply the annotation to the indicated selectid id, if there is an
// existing annotation it will be deleted
// [ ] [ ]
status scIMPL_EXPORT    SCSEL_ApplyAnnotation( scSelection*,
                                               const scAnnotation&,
                                               scRedispList* );

#endif

// ============================================================== */

void scIMPL_EXPORT      SCCOL_InvertExtents( scColumn*,
                                             HiliteFuncPtr,
                                             APPDrwCtx );
#if SCDEBUG > 1

// these return the size of the memory image of these structures
long scIMPL_EXPORT      SCCOL_DebugSize( scColumn* );
long scIMPL_EXPORT      SCSTR_DebugSize( scStream* );



void scIMPL_EXPORT      SCCOL_InvertRegion( scColumn*,
                                            HiliteFuncPtr,
                                            APPDrwCtx );

void scIMPL_EXPORT      SCDebugColumnList( void );
void scIMPL_EXPORT      SCDebugColumn( scColumn*,
                                       int  contentLevel );
```

```
                                        SubstituteFunc  func,
                                        scRedispList*   damage );


/*==================================================*/
// DEPRECATED
// check spelling in the stream
//
// [ ] [ ]
status scIMPL_EXPORT    SCSTR_Iter( scStream*       streamID,
                                    SubstituteFunc  func,
                                    scRedispList*   damage );


/*==================================================*/
// DEPRECATED
// [ ] [ ]
status scIMPL_EXPORT    SCSTR_Search( scStream*     streamID,
                                      const UCS2*       findString,
                                      SubstituteFunc    func,
                                      scRedispList* damage );

// @func Search for the string from the current selection. When
// the string is found move the selection to it.
status scIMPL_EXPORT    SCSEL_FindString(
                                scSelection*    select,     // @parm <c scSelection>
                                const UCS2* findString );   // @parm <t UCS2> string to find.

/*==================================================*/
// this returns whether or not the column potentially has text, if
// because of reformatting nothing lands in here we will still return
// true successful == has text
//


// [ ] [ ]
status scIMPL_EXPORT    SCCOL_HasText( scColumn* colID );

/*==================================================*/
// at start up this will have the first token selected

class stTokenIter {
public:
    virtual void    release() = 0;
    virtual void    reset() = 0;

                // sets a selection of the iterators para
    virtual int     paraselection( scSelection* ) = 0;

                // sets selection for the token iterator
    virtual int     setselection( scSelection* ) = 0;

                // retrieves the next token,
                // the string should have its size set in the
                // len field, if the token will not fit
                // in the string, a negative number is returned
                // that specifies the size, try again with a sufficient
                // size string
    virtual int     gettoken( stUnivString& ) = 0;

                // replaces the current token
    virtual int     replacetoken( stUnivString& ) = 0;

                // moves to next token
    virtual int     next() = 0;
};

class stContUnitIter {
public:
    virtual void    release() = 0;
    virtual void    reset() = 0;
    virtual int     gettokeniter( stTokenIter*& ) = 0;
    virtual int     next() = 0;
```

```
// @func Copies a selection of text, returning the copy in the scScrapPtr.
// <a name="SCSEL_CopyText">-</a>
status scIMPL_EXPORT    SCSEL_CopyText(
                            scSelection*    sel,        // @parm <c scSelection>
                            scScrapPtr&     scrap );    // @parm <c scScrapPtr>



// @func Pastes a selection of text contained in scScrapPtr into a stream
// at the insertion point marked by scSelection. If scSelection is
// a selection of one or more characters, the selected characters
// are deleted from the stream before the scScrapPtr text is pasted in.
// This operation copies the text as it pastes it. Therefore,
// multiple pastes can be made with the same scScrapPtr
// without making any explicit copies.
// @xref <f SCSEL_CutText>, <f SCSEL_CopyText>
status scIMPL_EXPORT    SCSEL_PasteText(
                            scSelection*    sel,    // @parm <c scSelection>
                            scScrapPtr      scrap,  // @parm <c scScrapPtr>
                            TypeSpec        ts,     // @parm If NULL uses prev spec.
                            scRedispList*   rInfo );// @parm <c scRedispList>
                                                    // Redisplay info, arg may be zero.



/*======================================================================*/
// @func The following is a useful call to get a stream from a selection and the
// the first typespec in the selection
//
status scIMPL_EXPORT    SCSEL_GetStream(
                            const scSelection*  sel,    // @parm <c scSelection>
                            scStream*&          str,    // @parm <c scStream>
                            TypeSpec&           ts );   // @parm The first <t TypeSpec>.



/*======================================================================*/
// see doc in html file

status SCSEL_InsertField( scSelection*,
                          const clField&,
                          TypeSpec&,
                          scRedispList* damage );


/*======================================================================*/
// The following are for spell checkings */

// actual definition in SCTYPES.H
//  Substitution function
//  inWord and outWord are null terminated strings
//  outWord is allocated and freed by application
//
// the substitution should function returns
//  - successful    if the word has been changed
//  - noAction      if no change is necessary
//  - other error   to be propogated back to app
//
//  typedef status (*SubstituteFunc)( UCS2**outWord, UCS2*inWord );
//
//

/*=============================================*/
// DEPRECATED
// check spelling in selection,
// this is best used in conjunction with SCMoveSelect
//
// [ ] [ ]
status scIMPL_EXPORT    SCSEL_Iter( scSelection*        selectID,
                                                        // to follow.
                          scKeyRecord*    keyRecs,    // @parm <c scKeyRecord>
                          scRedispList*   rInfo );    // @parm <c scRedispList>
                                                        // Redisplay info, arg may be zero.

// @func For immediate redisplay of text that has been altered in editing.
// It redisplays only those lines which the cursor has been on.
// Typing a carriage return should force two lines to be redisplayed
// immediately; likewise for a backspace at the beginning of a line.
// Normally only one line needs to be redisplayed.
// IF THE OPERATION CROSSES COLUMNS, ONLY THE COLUMN
// IN WHICH THE CURSOR ENDS UP IS UPDATED.
//
status scIMPL_EXPORT    SCCOL_UpdateLine(
                          scColumn*             col,        // @parm <c scSelection>
                          scImmediateRedisp&  immred,     // @parm <c scImmediateRedisp>
                          APPDrwCtx             drwctx );   // @parm <t APPDrwCtx>




// @func Applies a <t TypeSpec> to a <c scSelection>. The TypeSpec replaces
// all specs contained in the selection.
//
status scIMPL_EXPORT    SCSEL_SetTextStyle(
                          scSelection*    sel,          // @parm <c scSelection>
                          TypeSpec        ts,           // @parm <t TypeSpec>
                          scRedispList*   rInfo );      // @parm <c scRedispList>
                                                        // Redisplay info, arg may be zero.






// Applies a character transformation (to all caps, for example)
// to the selection.
//
// [ ] [ ]
status scIMPL_EXPORT    SCSEL_TextTrans( scSelection*,
                                         eChTranType,
                                         int,
                                         scRedispList* );





/*============ TEXT LEVEL CUT, COPY, PASTE & CLEAR =============*/
// These routines cut and paste paragraphs and text characters.
// All relevant paragraph attributes and character attributes are maintained.
// The scrap is maintained in internal Toolbox formats.
//

// @func Cuts a selection of text, returning it in the scScrapPtr.
//
status scIMPL_EXPORT    SCSEL_CutText(
                          scSelection*    sel,          // @parm <c scSelection>
                          scScrapPtr&     scrapPtr,     // @parm <c scScrapPtr>
                          scRedispList*   rInfo );      // @parm <c scRedispList>
                                                        // Redisplay info, arg may be zero.



// @func Deletes a selection of text.
//
status scIMPL_EXPORT    SCSEL_ClearText(
                          scSelection*    sel,          // @parm <c scSelection>
                          scRedispList*   rInfo );      // @parm <c scRedispList>
                                                        // Redisplay info, arg may be zero.
```

```
status scIMPL_EXPORT    SCSEL_Restore(
                            const scStream*          stream, // @parm <c scStream>
                            const scStreamLocation& mark,   // @parm <c scStreamLocation>
                            const scStreamLocation& point,  // @parm <c scStreamLocation>
                            scSelection*&            sel,    // @parm <c scSelection>
                            Bool                     geometryChange );   // has the layout changed



/*======================================================================*/
// @func Using a hitpt and a modifier (such as selection of a word or paragraph),
// returns a selection.
//
status scIMPL_EXPORT    SCCOL_SelectSpecial(
                            scColumn*           col,    // @parm <c scColumn>
                            const scMuPoint&    hitPt,  // @parm Hit point.
                            eSelectModifier     mod,    // @parm <t eSelectModifier>
                            scSelection*&       sel );  // @parm <c scSelection>



/*======================================================================*/
// This grows the selection according to the enum eSelectMove found in
// sctypes.h
//

status scIMPL_EXPORT    SCSEL_Move( scSelection*,
                                    eSelectMove );


/*======================================================================*/
// This alters the selection point according to the enum eSelectMove found in
// sctypes.h
//

status scIMPL_EXPORT    SCSEL_Extend( scSelection*, eSelectMove );


/*======================================================================*/
// select the nth pagraph of a stream - if you go off the end
// status returns noAction
//
// [ ] [ ]
status scIMPL_EXPORT    SCSTR_NthParaSelect( scStream*      streamID,
                                             long           nthPara,
                                             scSelection*   select );



/*======================================================================*/
// @func Highlights the current selection, using the function pointer passed
// in, the coordinates that will be contained in the call back are in
// object coordinates and MUST be transformed to device coordinates.
//
status scIMPL_EXPORT    SCSEL_Hilite(
                            scSelection*    sel,            // @parm <c scSelection>
                            HiliteFuncPtr   appDrawRect );  // @parm <t HiliteFuncPtr>


/*===================== EDITING MESSAGES =====================*/

// @func Inserts keystrokes into the given stream. Place one or more keystrokes
// into the array. NULL values in the key record array will simply be ignored.
// scKeyRecord* points to an array of keystrokes; numKeys is the number
// of elements in the array.
// Stores information for undo in the key records. Sample code will illustrate
// how to take the inverse values in scKeyRecord and use them to undo the
// insertion of keystrokes.
//
status scIMPL_EXPORT    SCSEL_InsertKeyRecords(
                            scSelection*    sel,        // @parm <c scSelection>
                            short           numRecs,    // @parm Number of key recs
                            const scMuPoint&    hitpt,  // @parm The hit point in
                                                        // object coordinates.
                            HiliteFuncPtr       hlfunc, // @parm <t HiliteFuncPtr>
                            APPDrwCtx           drwctx, // @parm <t APPDrwCtx>
                            scSelection*&       sel );  // @parm <c scSelection> to be filled in
                                                        // by the Composition Toolbox.



// @func This extends the selection from the scStreamLocation and then may
// be followed with <f SCCOL_ExtendSelect>.
// @xref <f SCCOL_StartSelect>
status scIMPL_EXPORT    SCCOL_StartSelect(
                            scColumn*           col,    // @parm <c scColumn>
                            scStreamLocation&   stloc,  // @parm <c scStreamLocation>
                            const scMuPoint&    hitpt,  // @parm The hit point in
                                                        // object coordinates.
                            HiliteFuncPtr       hlfunc, // @parm <t HiliteFuncPtr>
                            APPDrwCtx           drwctx, // @parm <t APPDrwCtx>
                            scSelection*&       sel );  // @parm <c scSelection>



// @func Extends a selection derived from <f SCCOL_ExtendSelect>. Can
// be used over multiple columns in a flowset.

status scIMPL_EXPORT    SCCOL_ExtendSelect(
                            scColumn*           col,    // @parm <c scColumn>
                            const scMuPoint&    hitpt,  // @parm The hit point in
                                                        // object coordinates.
                            HiliteFuncPtr       hlfunc, // @parm <t HiliteFuncPtr>
                            APPDrwCtx           drwctx, // @parm <t APPDrwCtx>
                            scSelection*        sel );  // @parm <c scSelection>



// @func Converts a <c scSelection> into a mark and point.
// The mark is guaranteed to logically precede the point.
// This call is typically used in conjunction with <f SCSEL_Restore> and
// to determine information at the selection point.
// @xref <f SCCOL_StartSelect>, <f SCCOL_ExtendSelect>, <f SCCOL_InitialSelect>

status scIMPL_EXPORT    SCSEL_Decompose(
                            scSelection*        sel,    // @parm <c scSelection>
                            scStreamLocation&   mark,   // @parm <c scStreamLocation>
                            scStreamLocation&   point );// @parm <c scStreamLocation>

// same as above except that the selection is not sorted
status scIMPL_EXPORT    SCSEL_Decompose2(
                            scSelection*        sel,    // @parm <c scSelection>
                            scStreamLocation&   mark,   // @parm <c scStreamLocation>
                            scStreamLocation&   point );// @parm <c scStreamLocation>


// @func Invalidates the selection in the toolbox, changes the selection
// to null and invalidates the selection in the toolbox

status scIMPL_EXPORT    SCSEL_Invalidate(
                            scSelection*& sel );    // @parm <c scSelection>

/*=================================================================*/
// @func Sets up a text selection, using the given
// mark and point. Useful for restoring the selection
// when re-activating a document, and for undo and redo,
// especially for undoing arrow and backspace keystrokes.
// In this call the <c scStreamLocation> need only have the following
// member variables filled in:
// <nl> fParaNum
// <nl> fParaOffset
// <nl>All the rest are unneeded. After this call subsequent calls
// to <f SCSEL_Decompose> will fill in the scStreamLocation values
// correctly.
```

```
status scIMPL_EXPORT    SCOBJ_PtrRestore(
                            scTBObj*    obj,          // @parm Restore this objects pointers.
                            scSet*      enumTable );  // @parm Use this enumtable.


// @func Prior to calling SCOBJ_PtrRestore the client may want to abort
// the action for some reason. In that case call the following
// and all objects that have been read in, but have not had
// their pointers restored will be deleted, including the
// the enumeration table (scSet)
//
status scIMPL_EXPORT    SCSET_Abort(
                            scSet*& enumTable );   // @parm <c scSet>


// @func Gives the size of a Toolbox object that will be written to disk.
// The first column in a flow set will contain the content/sctream.
//
status scIMPL_EXPORT    SCExternalSize(
                            scColumn*   col,        // @parm <c scColumn>
                            long&       bytes );    // @parm Disk bytes.


/*==================== SELECTION MESSAGES ====================*/


// @func Forces the initial selection within an empty container by creating
// an initial stream. Two conditions present interesting error conditions
// with this call.
// <nl>1. If the formatted text cannot fit into the container a
// scERRstructure is returned,
// <nl>2. If the container is not the first in a flow set then
// scERRlogical is returned.
// <nl>The client must perform the first highlighting of the cursor by
// following this call with a call to <f SCHiLite>.
//
status scIMPL_EXPORT    SCCOL_InitialSelect(
                            scColumn*       col,        // @parm <c scColumn>
                            TypeSpec&       spec,       // @parm <t TypeSpec>
                            scSelection*&   select );   // @parm <c scSelection>




// @func Provides information on how good the hit is,
// to be used for selection evaluation.
// the REAL num is the distance squared in micropoints from the
// hitpoint to the nearest charcter and its baseline
//
status scIMPL_EXPORT    SCCOL_ClickEvaluate(
                            scColumn*           col,   // @parm <c scColumn>.
                            const scMuPoint& evalpt,// @parm Point to evaluate.
                            REAL&               dist );// @parm Squared dist in <t MicroPoints>.


// @func The mouse down click should force a call of SCCOL_StartSelect and
// mouse moves should get <f SCCOL_ExtendSelect>. Effectively the StartSelect
// message forces a flow set to get the focus. Is is an error to call
// ExtendSelect with a column from a different flow set than was called
// from the original StartSelect. Also the client may want to coerce points
// to lie within the column's extents. If the container is rotated the coercion
// should happen in the containers coordinate space to insure correct
// interpretation of the coercion.
// After the first StartSelect message the Selection is accurate so if
// auto scrolling is necessary it may be done, provided the clip region
// is set up correctly.
// <nl>
// <nl>NOTE: The caller should filter out redundant mouse hits, (i.e if the
// current mouse hit is the same as the last mouse hit don't call
// SCCOL_ExtendSelect
//
status scIMPL_EXPORT    SCCOL_StartSelect(
                            scColumn*           col,    // @parm <c scColumn>
```

```
// These routines read and write the Toolbox structures to disk. */

// @func Tells the Toolbox that the application is about to
// commence writing structures out. This zeros the enumeration
// count of all objects in the Toolbox within this context
//
status scIMPL_EXPORT    SCTB_ZeroEnumeration( void );



// @func Tells the object to enumerate itself.
//
status scIMPL_EXPORT    SCOBJ_Enumerate(
                            scTBObj*    obj,
                            long&       ecount );


// @func Stores the structures for this object. Streams and linked columns
// are by default written out with the first column; writes to columns
// that are not the first column of a stream are no-ops.
//
status scIMPL_EXPORT    SCCOL_Write(
                            scColumn*   col,    // @parm <c scColumn>
                            APPCtxPtr   ctxp,   // @parm <t APPCtxPtr>
                            IOFuncPtr   write );// @parm <t IOFuncPtr>



// @func Tells the Toolbox that the application is about to
// commence reading structures in. This allocates an enumeration
// structure of class scSet, when the file was written out the
// client probably should have noted the enumeration count, by doing this
// we may allocate enough members of the enumeration structure
// to at least guarantee that there will be no failure in inserting
// members into the enumeration structure
//
status scIMPL_EXPORT    SCSET_InitRead(
                            scSet*& enumerationTable,       // @parm Pointer to enum table
                                                            // that Composition Toolbox will
                                                            // allocate - pre allocating the
                                                            // number of slots indicated to
                                                            // minimize memory failures on
                                                            // file i/o.
                            long    preAllocationCount );   // @parm Preallocate this many slots
                                                            // in the enum table.

// @func Lets the Toolbox know the application is finished reading
// so it can free structures for restoring pointers and will
// recompose everything that needs recompostion.
//
status scIMPL_EXPORT    SCSET_FiniRead(
                            scSet*          enumTable,  // @parm This table will be freed.
                            scRedispList*   rInfo );    // @parm <c scRedispList>
                                                        // Redisplay info, arg may be zero.



// @func Read this column.
//
status scIMPL_EXPORT    SCCOL_Read(
                            APPColumn   appcol,     // @parm Client's <t APPColumn> to
                                                    // associate with this column.
                            scColumn*&  col,        // @parm <c scColumn>
                            scSet*      enumTable,  // @parm <c scSet>
                            APPCtxPtr   ioctxptr,   // @parm <t APPCtxPtr> Abstract file i/o type.
                            IOFuncPtr   ioFuncPtr );// @parm <t IOFuncPtr> write function pointer.


// @func Tells the object to restore its pointers. This
// function relies upon <f APPDiskIDToPointer>.
//
```

```
                                scSelection*        sel,        // @parm <c scColumn> in flow set to pas
te
                                                                // the text.
                                stTextImportExport&     appText,// @parm <c scAPPText> contains "marked
up" text.
                                scRedispList*   rInfo );    // @parm <c scRedispList>
                                                            // Redisplay info, arg may be zero.


// @func Returns a copy of the given stream in APPText&.
// @xref <f SCFS_PasteAPPText>

status scIMPL_EXPORT    SCSTR_GetAPPText(
                                scStream*       str,        // @parm <c scStream> to get "marked up" tex
t from.
                                stTextImportExport&     appText );  // @parm <c scAPPText> contains "mar
ked up" text.

status scIMPL_EXPORT    SCSEL_GetAPPText(
                                scSelection*            str,        // @parm <c scStream> to get "marked
 up" text from.
                                stTextImportExport&     appText );  // @parm <c scAPPText> contains "mar
ked up" text.


/*======================== CONTENT I/O ========================*/

// These routines read and write ASCII text files with mark-up. */

// @func Imports Latin-1 text -- adds the contents of the text file
// to the column using the TypeSpec as the default text specification.
// The call back IO function should conform to the header:

status scIMPL_EXPORT    SCFS_ReadTextFile(
                                scColumn*       col,    // @parm <c scColumn>
                                TypeSpec        spec,   // @parm <t TypeSpec>
                                APPCtxPtr       ctxp,   // @parm <t APPCtxPtr>
                                IOFuncPtr       read,   // @parm <t IOFuncPtr>
                                scRedispList*   rInfo );    // @parm <c scRedispList>
                                                            // Redisplay info, arg may be zero.



// @func Exports text -- writes the stream to the text file.
//
status scIMPL_EXPORT    SCSTR_WriteTextFile(
                                scStream*   stream, // @parm <c scStream>
                                APPCtxPtr   ctxp,   // @parm <t APPCtxPtr>
                                IOFuncPtr   write );// @parm <t IOFuncPtr>


// @func Reads an Latin-1 text file and returns a scrap handle to it.
// This is useful for importing text and pasting it into a stream
// at an insertion point. The call back IO function should
// conform to the same header as above.
//
status scIMPL_EXPORT    SCSCR_TextFile(
                                scScrapPtr&     scrapP, // @parm <t scScrapPtr>
                                APPCtxPtr       ctxp,   // @parm <t APPCtxPtr>
                                IOFuncPtr       read ); // @parm <t IOFuncPtr>



/*========================  FILE I/O ========================*/
```

```
                              scSpecLocList&  cslist );   // @parm <c scCharSpecList> has
                                                          // positions of all specs contained.

status scIMPL_EXPORT    SCSEL_PARATSList( scSelection*     sel,
                                          scSpecLocList&   cslist );



// @func Counts the characters in a stream. This
// does not repesent an exact count for file i/o
// of characters written out.
//

status scIMPL_EXPORT    SCSTR_ChCount(
                              scStream*   str,        // @parm <c scStream> to query.
                              long&       chCount );  // @parm Characters in stream.




#ifdef scFlowJustify

// @func Sets the vertical justification attributes for the column to be
// flush top (no justification), flush bottom, centered, justified,
// or force justified.
//
status scIMPL_EXPORT    SCCOL_FlowJustify(
                              scColumn*   col,        // @parm <c scColumn>
                              eVertJust   vj );       // @parm <t eVertJust>



// @func Sets the depth of a vertically flexible column and
// vertically justifies it. This is the only way to vertically
// justify a vert flex column. It should not be used with columns
// that are not vert flex.
// @xref <f SCCOL_SetVertFlex>

status scIMPL_EXPORT    SCCOL_SetDepthNVJ(
                              scColumn*       col,        // @parm <c scColumn>
                              MicroPoint      depth,      // @parm Depth to VJ to.
                              scRedispList*   rInfo );    // @parm <c scRedispList>
                                                          // Redisplay info, arg may be zero.




#endif /* scFlowJustify */


/*============== CONTENT CUT, COPY, PASTE & CLEAR ==============*/


// These routines move text in and out of the Toolbox,
// with filters converting text as necessary.
//

// @func Appends the text contained in the APPText to the end
// of the stream associated with the scColumn* (formerly SCReadAPPText).
// @xref <f SCSTR_GetAPPText>

status scIMPL_EXPORT    SCFS_PasteAPPText(
                              scColumn*       col,        // @parm <c scColumn> in flow set to paste
                                                          // the text.
                              stTextImportExport&     appText,    // @parm <c scAPPText> contains "mar
ked up" text.
                              scRedispList*   rInfo );    // @parm <c scRedispList>
                                                          // Redisplay info, arg may be zero.


status scIMPL_EXPORT    SCSEL_PasteAPPText(
                                               // reformatted to reflect the change.
                          eSpecTask      specTask,   // @parm <t eSpecTask> tells the toolbox
                                               // what action to take to repair the change.
                          scRedispList*  rInfo );    // @parm <c scRedispList>
                                               // Redisplay info, arg may be zero.




// @func Gets a list of TypeSpecs in a column.
// @xref <f SCSTR_TSList>, <f SCSEL_TSList>

status scIMPL_EXPORT    SCCOL_TSList(
                          scColumn*       col,         // @parm <c scColumn> to query.
                          scTypeSpecList& tslist );    // @parm <c scTypeSpecList> contains
                                                       // a list of specs used.

// @func Gets a list of TypeSpecs in a stream. When working with linked columns,
// this is more efficient than iteratively calling SCColTSList.
// @xref <f SCCOL_TSList>, <f SCSEL_TSList>

status scIMPL_EXPORT    SCSTR_TSList(
                          scStream*       col,         // @parm <c scStream> to query.
                          scTypeSpecList& tslist );    // @parm <c scTypeSpecList> contains
                                                       // a list of specs used.

// @func Gets a list of ParaTypeSpecs in a stream.
// @xref <f SCCOL_TSList>, <f SCSEL_TSList>

status scIMPL_EXPORT    SCSTR_ParaTSList(
                          scStream*       col,         // @parm <c scStream> to query.
                          scTypeSpecList& tslist );    // @parm <c scTypeSpecList> contains
                                                       // a list of specs used.

// @func Gets a list of TypeSpecs in a selection.
// @xref <f SCCOL_TSList>, <f SCSTR_TSList>

status scIMPL_EXPORT    SCSEL_TSList(
                          scSelection*    sel,         // @parm <c scSelection>.
                          scTypeSpecList& tslist );    // @parm <c scTypeSpecList> contains
                                                       // a list of specs used.
status scIMPL_EXPORT    SCSEL_PARATSList(
                          scSelection*    sel,         // @parm <c scSelection>.
                          scTypeSpecList& tslist );    // @parm <c scTypeSpecList> contains
                                                       // a list of specs used.


// @func Gets a list of the (TypeSpec, character location) pairs
// representing the TypeSpec runs of the stream.

status scIMPL_EXPORT    SCSTR_CHTSList(
                          scStream*        stream,     // @parm <c scStream> to query.
                          scSpecLocList&  cslist );    // @parm <c scCharSpecList> has
                                                       // positions of all specs contained.

status scIMPL_EXPORT    SCSTR_PARATSList( scStream*        stream,
                                          scSpecLocList&   cslist );


// @func Gets a list of the (TypeSpec, character location) pairs
// representing the TypeSpec runs of the selection.
// CAUTION: the ends of paragraphs are marked by NULL specs,
// so depending on how many paragraphs the selection traverses,
// there may be multiple NULL specs contained in the list.
// These NULL specs are not terminators of the list;
// rely upon the CharSpecListHandle's count field for the
// accurate number of structures in the list.
//

status scIMPL_EXPORT    SCSEL_CHTSList(
                          scSelection*     sel,        // @parm <c scSelection> to query.
```

```
                            TypeSpec    ts );       // @parm <t TypeSpec> to apply to string.



// @func Frees the scScrapPtr.
//
status scIMPL_EXPORT    SCSCR_Free(
                            scScrapPtr scrap );     // @parm <c scScrapPtr>



// @func Returns the list of specs referenced by the contents of the scScrapPtr.
//
status scIMPL_EXPORT    SCSCR_TSList(
                            scScrapPtr      scrap,      // @parm <c scScrapPtr>
                            scTypeSpecList& tslist );   // @parm <c scTypeSpecList>



// @func Writes the contents of the scScrapPtr to disk.

status scIMPL_EXPORT    SCSCR_Write(
                            scScrapPtr  scrapPtr,       // @parm <c scScrapPtr>
                            APPCtxPtr   ctxPtr,         // @parm <t APPCtxPtr>
                            IOFuncPtr   iofuncp );      // @parm <t IOFuncPtr>



// @func Reads from disk into the scScrapPtr.
// Scrap to be read must be a column.
// @xref <f SCSET_InitRead>
status scIMPL_EXPORT    SCSCR_ReadCol(
                            scScrapPtr&     scrap,      // @parm <c scScrapPtr>
                            scSet*          enumtable,  // @parm <c scSet>
                            APPCtxPtr       ctxPtr,     // @parm <t APPCtxPtr>
                            IOFuncPtr       readFunc ); // @parm <t IOFuncPtr>



// @func Reads from disk into the scScrapPtr, using the call back
// read routine, which should conform to the same header as above.
// Scrap to be read must be a stream.
//
status scIMPL_EXPORT    SCSCR_ReadStream(
                            scScrapPtr&     scrap,      // @parm <c scScrapPtr>
                            scSet*          enumtable,  // @parm <c scSet>
                            APPCtxPtr       ctxPtr,     // @parm <t APPCtxPtr>
                            IOFuncPtr       readFunc ); // @parm <t IOFuncPtr>



/*==================== TYPE SPECIFICATION ====================*/


// @func Informs the Toolbox that the TypeSpec has been changed and
// tells it what action needs to be taken to respond accordingly.
// The Toolbox will recompose, rebreak, or repaint (as instructed
// by SpecTask) and report back on damage. SpecTasks may be ORed
// together to indicate multiple tasks. The host application
// can derive the tasks by calling the function SpecTaskCalculate,
// located in the source module (delivered with the Toolbox)
// sc_spchg.cpp.


status scIMPL_EXPORT    SCENG_ChangedTS(
                            TypeSpec        ts,         // @parm <t TypeSpec> has changed and text n
eeds to be
```

```
// in the column's coordinate system.
//
status scIMPL_EXPORT    SCCOL_QueryInkExtents(
                            scColumn*   col,            // @parm <c scColumn> to query.
                            scXRect&    inkExtents );   // @parm <t scXRect>


// @func Queries margins of the column. Returns the actual size of the
// container, rather than the extents. Only meaningful for rectangular
// containers. Bear in mind that the ink of text may extend outside of the
// container.
//
status scIMPL_EXPORT    SCCOL_QueryMargins(
                            scColumn*   col,        // @parm <c scColumn> to query.
                            scXRect&    margins );  // @parm <t scXRect>


// Queries positions of the column's lines. For each line, it queries
// the baseline, the extents, and the character characteristics at
// the start (x height, ascender height, descender height, cap height, etc. ).
// The last two parameters indicate the number of lines and whether text
// overflows the column.
// Typically used for alignment purposes. Positions are in the container's
// local coordinates.
//
// [ ] [ ]
status scIMPL_EXPORT    SCCOL_LinePositions( scColumn*,
                                             scLineInfoList*,
                                             long&,
                                             Bool& );


// @func Queries the depth of the column. Useful for determining the depth
// of irregularly shaped columns.
//
status scIMPL_EXPORT    SCCOL_Size(
                            scColumn*   col,    // @parm <c scColumn>
                            scSize&     size ); // @parm <c scSize>


/*==================== SCRAP CONVERSION  =====================*/

// These routines provide a means of converting between the Toolbox
// world and the outside world, typically a conversion into the
// lowest common denominator -- text.
//


// @func Converts Toolbox scrap in the scScrapPtr to global scrap and
// stores it in the given Handle, which should be passed in as a
// valid handle. The Handle then contains
// a "C" string. This is one case where the Toolbox will not free
// the new structure it creates.
//
status scIMPL_EXPORT    SCSCR_ConToSys(
                            scScrapPtr          scrap,      // @parm <c scScrapPtr>
                            SystemMemoryObject& memobj );   // @parm <c SystemMemoryObject>



// @func Converts global scrap in the Handle to Toolbox scrap
// and places it in the given scScrapPtr, putting it on
// the internal Toolbox clipboard. The handle should contain
// a "C" string, so that when 0 is encountered, we stop reading.
//
status scIMPL_EXPORT    SCSCR_SysToCon(
                            scScrapPtr& scrap,      // @parm <c scScrapPtr>

                            const scChar*   stringscrap,// @parm Null terminated "C" string
                                                // from system scrap.
                            const scXRect&   rect );    // @parm <c scXRect>


// Perform Boolean operations on regions, the 3rd arg may be one of the
// 2 original regions, in that case it will replace the contents of after the
// operation is complete.

// @func Performs an intersection of two regions, placing the intersection
// in a third region.

status scIMPL_EXPORT    SCHRGN_Sect(
                            const HRgnHandle r1,            // @parm <t HRgnHandle>
                            const HRgnHandle r2,            // @parm <t HRgnHandle>
                            HRgnHandle       intersection );// @parm <t HRgnHandle>,
                                                            // the intersection of r1 & r2.

// @func Performs the union of two regions, placing the union in a third region.

status scIMPL_EXPORT    SCHRGN_Union(
                            const HRgnHandle r1,            // @parm <t HRgnHandle>
                            const HRgnHandle r2,            // @parm <t HRgnHandle>
                            HRgnHandle       rUnion );      // @parm <t HRgnHandle>,
                                                            // the union of r1 & r2.


// @func Performs a difference of two regions, placing the diff
// in a third region.

status scIMPL_EXPORT    SCHRGN_Diff(
                            const HRgnHandle r1,            // @parm <t HRgnHandle>
                            const HRgnHandle r2,            // @parm <t HRgnHandle>
                            HRgnHandle       difference );  // @parm <t HRgnHandle>,
                                                            // the difference of r1 & r2.


// @func Performs an xor of two regions, placing the result in a third region.

status scIMPL_EXPORT    SCHRGN_Xor(
                            const HRgnHandle r1,            // @parm <t HRgnHandle>
                            const HRgnHandle r2,            // @parm <t HRgnHandle>
                            HRgnHandle       xor );         // @parm <t HRgnHandle>,
                                                            // the xor of r1 & r2.


#endif


/*======================= RENDERING ========================*/


// @func Renders/draws that part of the column lying within the
// given rect. The scXRect is in local coordinates. The Toolbox
// then calls back to the client using <f APPDrawStartLine>,
// <f APPDrawString>, & <f APPDrawEndLine>, passing the <t APPDrwCtx>
// through. This call and <f SCCOL_UpdateLine> are the only two calls
// that cause glyphs to be drawn. ALL DRAWING OF TOOLBOX CONTAINERS
// HAPPENS AT THE  BEHEST OF THE CLIENT.
// @xref <k APPDrawStartLine>, <k APPDrawString>, & <k APPDrawEndLine>

status scIMPL_EXPORT    SCCOL_Update(
                            scColumn*        col,        // @parm <c scColumn> to draw
                            const scXRect&   clipRect,   // @parm Clip rect.
                            APPDrwCtx        dc  );      // @parm Drawing context.



/*================= CONTAINER & LINE EXTENTS =================*/


// @func Queries ink extents of the column. The extents returned are
// the maximum bounding box of the maximum character extents expressed
```

```
                                const HRgnHandle rgn,          // @parm <t HRgnHandle>
                                MicroPoint&      sliverSize);  // @parm Size of sliver.

// @func Dispose a region.

status scIMPL_EXPORT    SCHRGN_Dispose(
                                HRgnHandle disRgn );           // @parm <t HRgnHandle> to dispose.

// @func Make a region empty, remove all slivers.

status scIMPL_EXPORT    SCHRGN_SetEmpty(
                                HRgnHandle emptyRgn );         // @parm <t HRgnHandle> to empty.

// @func Is a region empty.
// @rdesc scSuccess == empty region

status scIMPL_EXPORT    SCHRGN_Empty(
                                const HRgnHandle emptyRgn );   // @parm <t HRgnHandle> to test.

// @func Compare to regions for equality.
// @rdesc scSuccess == equality

status scIMPL_EXPORT    SCHRGN_Equal(
                                const HRgnHandle rgn1,     // @parm <t HRgnHandle>.
                                const HRgnHandle rgn2 );   // @parm <t HRgnHandle>.

// @func Determine if point is in region.
// @rdesc scSuccess == equality
//
status scIMPL_EXPORT    SCHRGN_PtIn(
                                const HRgnHandle    rgn,        // @parm <t HRgnHandle> to test.
                                const scMuPoint&    pt );       // @parm <c scMuPoint>

// @func Make a region rectangular.
//
status scIMPL_EXPORT    SCHRGN_Rect(
                                HRgnHandle      rng,    // @parm Region to apply rect.
                                const scXRect&  rect ); // @parm <c scXRect>

// @func Make a region from a set of verticies ( closed polygons(s) ).
// The polygon must have both a horizontal and vertical dimension
// (e.g. it must have interior space)

status scIMPL_EXPORT    SCHRGN_Poly(
                                HRgnHandle      rng,    // @parm Region to apply polygon.
                                const scVertex* polys );// @parm <c scVertex> Polygon(s) description.

// @func Copy a region.

status scIMPL_EXPORT    SCHRGN_Copy(
                                HRgnHandle       dstRgn,    // @parm The copy.
                                const HRgnHandle srcRgn );  // @parm To copy.

// @func Translate a region.

status scIMPL_EXPORT    SCHRGN_Translate(
                                HRgnHandle  rgn,    // @parm Region to inset.
                                MicroPoint  x,      // @parm Horizontal translation.
                                MicroPoint  y );    // @parm Vertical translation.


// @func Inset a region.

status scIMPL_EXPORT    SCHRGN_Inset(
                                HRgnHandle  rgn,    // @parm Region to inset.
                                MicroPoint  h,      // @parm Horizontal size change.
                                MicroPoint  v );    // @parm Vertical size change.


// @func Is this rect contained within the region.

status scIMPL_EXPORT    SCHRGN_RectIn(
                                const HRgnHandle rgn,       // @parm Region to test.
```

```
// vertex list.
//
// [ ] [ ]
status scIMPL_EXPORT    SCCOL_PastePoly( scColumn*,
                                 const scVertex*,
                                 scRedispList* );


// Extracts a copy of the polygon applied to this column,
// causing no recomposition. Useful for editing the polygon.
//
// [ ] [ ]
status scIMPL_EXPORT    SCCOL_CopyPoly( scColumn*,
                                scVertex*& );

// Clears the polygon applied to this column.
//
// [ ] [ ]
status scIMPL_EXPORT    SCCOL_ClearPoly( scColumn*,
                                 scRedispList* );



/*================ REGION RUN-AROUND OPERATIONS ================*/

// Regions are high precision descriptions of arbitrary shapes.
// They are useful for representing irregularly shaped containers,
// or for enabling text to run around objects intersecting a given container.
// The application may use them to represent containers
// that have been modified by intersection with other page objects.
//

// @func Applies a region as the shape of this column.
// The region is assumed to be in local coordinates.
//
status scIMPL_EXPORT    SCCOL_PasteRgn(
                            scColumn*           col,    // @parm <c scColumn> to apply region to.
                            const HRgnHandle    rgn,    // @parm Region to apply.
                            scRedispList*   rInfo );    // @parm <c scRedispList>
                                                        // Redisplay info, arg may be zero.

// @func Extracts a copy of the region applied to this column,
// causing no recomposition.
//
status scIMPL_EXPORT    SCCOL_CopyRgn(
                            scColumn*   col,    // @parm <c scColumn> with region
                            HRgnHandle& rgn );  // @parm <t HRgnHandle> the region copy.



// @func Clears the region belonging to this column.
//
status scIMPL_EXPORT    SCCOL_ClearRgn(
                            scColumn*       col,        // @parm <c scColumn> to clear region.
                            scRedispList*   rInfo );    // @parm <c scRedispList>
                                                        // Redisplay info, arg may be zero.



// The following functions operate on regions.

// @func Create a new region.
//

status scIMPL_EXPORT    SCHRGN_New(
                            HRgnHandle& newRgn,         // @parm <t HRgnHandle>
                            MicroPoint  sliverSize);    // @parm Sliver size.

// @func What sliver size is a region using.

status scIMPL_EXPORT    SCHRGN_SliverSize(
                            scRedispList*   rInfo );// @parm <c scRedispList>
                                                    // Redisplay info, arg may be zero.


// @func Gets the vert flex attribute of the column.
// @xref <f SCCOL_SetVertFlex>, <f SCCOL_ClearVertFlex>

status scIMPL_EXPORT    SCCOL_GetVertFlex(
                            scColumn*   col,        // @parm <c scColumn> to get attribute from.
                            Bool&       onOff );    // @parm Vertical flex attribute, true
                                                    // equals on, false off.




// @func Turns on horizontal flexibility.
// @xref <f SCCOL_GetHorzFlex>, <f SCCOL_ClearHorzFlex>

status scIMPL_EXPORT    SCCOL_SetHorzFlex(
                            scColumn*   col,    // @parm <c scColumn> to set flex.
                            scRedispList*   rInfo );// @parm <c scRedispList>
                                                    // Redisplay info, arg may be zero.



// @func Turns off horizontal flexibility.
// @xref <f SCCOL_SetHorzFlex>, <f SCCOL_GetHorzFlex>

status scIMPL_EXPORT    SCCOL_ClearHorzFlex(
                            scColumn*       col,    // @parm <c scColumn> to clear flex.
                            scRedispList*   rInfo );// @parm <c scRedispList>
                                                    // Redisplay info, arg may be zero.




// @func Gets the horzontal flex attribute of the column.
// @xref <f SCCOL_SetHorzFlex>, <f SCCOL_ClearHorzFlex>

status scIMPL_EXPORT    SCCOL_GetHorzFlex(
                            scColumn*   col,        // @parm <c scColumn> to get attribute from.
                            Bool&       onOff );    // @parm Horizontal flex attribute, true
                                                    // equals on, false off.

// @func Gets the direction of lines in the container,
// and of characters in the lines.
// @xref <f SCCOL_SetFlowDirection>

status scIMPL_EXPORT    SCCOL_GetFlowDirection(
                            scColumn*   col,        // @parm <c scColumn> to query.
                            scFlowDir&  fd );       // @parm <c scFlowDir> of column.

// @func Sets the direction of lines in the container,
// and of characters in the lines.
// @xref <f SCCOL_GetFlowDirection>

status scIMPL_EXPORT    SCCOL_SetFlowDirection(
                            scColumn*           col,    // @parm <c scColumn> to set.
                            const scFlowDir&    fd );   // @parm <c scFlowDir> value to
                                                        // set of column.

/*============= POLYGON CONTAINER SHAPE OPERATIONS =============*/

#if defined( scColumnShape )

// The application may create polygons in any way that it sees fit.
// They are then passed into the Toolbox and text flows into them
// using an even-odd area fill algorithm.
//


// Pastes in a set of vertices to be added to the column's current
```

```
status scIMPL_EXPORT    SCFS_Recompose(
                            scColumn*       col,    // @parm Flow set to recompose.
                            scRedispList*   rInfo );// @parm <c scRedispList>
                                                    // Redisplay info, arg may be zero.

// NOT IMPLEMENTED
// Recompose a portion of the stream in the flowset. Process the flowset
// for the number of ticks indicated. We will process paragraphs
// until time exceeds the ticks
//
// [ ] [ ]
status scIMPL_EXPORT    SCFS_Recompose( scColumn*,
                                        long ticks,
                                        scRedispList* );


// Rebreaks all the lines in a column, with extreme prejudice;
// it ignores the flowset RecomposeHold() setting and processes
// only the indicated column, it will return an error if a prior
// column is uncomposed
//
// [ ] [ ]
status scIMPL_EXPORT    SCCOL_Recompose( scColumn*,
                                         scRedispList* );


// @func Applies the specs in the CharSpecListHandle to the text
// at the locations indicated therein.
// @xref <f SCCOL_GetStream>, <f SCSTR_CHTSList>
status scIMPL_EXPORT    SCSTR_CHTSListSet(
                            scStream*       stream, // @parm <c scStream> to apply scCharSpecList to

                            const scSpecLocList& cslist,    // @parm <c scCharSpecList> list of spec
s and locations.
                            scRedispList*   rInfo );// @parm <c scRedispList>
                                                    // Redisplay info, arg may be zero.

status scIMPL_EXPORT    SCSTR_PARATSListSet( scStream*              stream,
                                             const scSpecLocList&   cslist,
                                             scRedispList*          rInfo );

/*============== CONTAINER CONSTRAINT OPERATIONS ==============*/


// These routines set containers to be flexible in the horizontal or
// vertical dimensions. A flexible container varies in size with its contents.
// A vertically flexible container varies with the number of lines;
// it grows until it reaches either the last character in the stream or
// a column break. A horizontally flexible container varies with the width
// of its widest line; it grows until the end of the paragraph or a hard
// return.
//
// NOTE: !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
// Container FLEXIBILITY and IRREGULAR shapes are MUTUALLY EXCLUSIVE!!!!!!
// Setting a container to be flexible automatically clears any
// irregular shape associated with the column.
//


// @func Turns on vertical flexibility.
// @xref <f SCCOL_GetVertFlex>, <f SCCOL_ClearVertFlex>

status scIMPL_EXPORT    SCCOL_SetVertFlex(
                            scColumn*       col,    // @parm <c scColumn> to set flex.
                            scRedispList*   rInfo );// @parm <c scRedispList>
                                                    // Redisplay info, arg may be zero.


// @func Turns off vertical flexibility.
// @xref <f SCCOL_SetVertFlex>, <f SCCOL_GetVertFlex>

status scIMPL_EXPORT    SCCOL_ClearVertFlex(
                            scColumn*       col,    // @parm <c scColumn> to clear flex.
```

```
// @func Copies a stream, returning a unique id in the second argument.
// The copy will not be associated with any containers.
// @xref <f SCCOL_GetStream>
//
status scIMPL_EXPORT    SCSTR_Copy(
                            const scStream* srcStream,  // @parm Stream to copy.
                            scStream*&      theCopy );  // @parm The new copy.




// @func Pastes a stream into a container. If the container already has a stream,
// the pasted stream is appended to the existing one. To conserve on
// resources, the streamID is not duplicated. Thus, for multiple
// pastes of one stream, a new copy of the stream must be made for each paste.
//
status scIMPL_EXPORT    SCFS_PasteStream(
                            scColumn*       col,         // @parm <c scColumn> containing stream
                                                         // which theStream will be appended.
                            scStream*       theStream,   // @parm The stream to be appended.
                            scRedispList*   rInfo );     // @parm <c scRedispList>
                                                         // Redisplay info, arg may be zero.


// Extracts a scContUnit from a scStreamLocation for use with SCSTR_Split
status scIMPL_EXPORT    SCSEL_GetContUnit(
                             scContUnit*& mark,
                             scContUnit*& point,
                             const scSelection* );

//
// This call is used to undo a link. To set this up,
// before linking two columns col1 and col2, save references
// to their streams, stream1 and stream2, respectively.
// When unlinking col1 and col2, call scStreamSplit( stream1, stream2 )
// to split the stream into its original two pieces. This call should be
// followed with a call to SCPasteStream( col2, stream2, scRedispList *)
// to reset the unlinked column's stream to its original value.
// NOTE: both stream1 and stream2 are assumed to be valid (non-NULL).
// This call should only be made with reformatting turned off.
//
// [ ] [ ]
status scIMPL_EXPORT    SCSTR_Split(
                            scStream*,
                            scContUnit*,
                            scStream*& );

//
// @func Compare streams for equality, this tests content and specs
// scSuccess == equality
// @xref <f SCCOL_GetStream>
//
status scIMPL_EXPORT    SCSTR_Compare(
                            const scStream* str1,       // @parm <c scStream>
                            const scStream* str2 );     // @parm <c scStream>


/*==== COMPOSITION OPERATIONS & COMPOSITION ERROR RECOVERY =====*/
// @func Sets recomposition off or on in the flow set of the indicated column
//
status scIMPL_EXPORT    SCFS_SetRecompose(
                            scColumn*   col,        // @parm Flow set to turn composition off or on
in.
                            Bool        onOff );    // @parm true == on, false = off

// @func Gets the current recomposition state of the flow set.
//
status scIMPL_EXPORT    SCFS_GetRecompose(
                            scColumn*   col,        // @parm Flow set to query.
                            Bool&       onOff );    // @parm Composition flag.

// @func Recomposes the flowset.
//
```

```
status scIMPL_EXPORT    SCCOL_SetSize(
                            scColumn*       col,        // @parm <c scColumn> to resize.
                            MicroPoint      width,      // @parm New <t MicroPoint> width.
                            MicroPoint      depth,      // @parm New <t MicroPoint> depth.
                            scRedispList*   rInfo );    // @parm <c scRedispList>
                                                        // Redisplay info, arg may be zero.

// tests to see if there is more text than is in this column
// this would set the flag to true if:
//          there is text in subsequent linked columns
//          there is unformatted text that will not fit in this column

status scIMPL_EXPORT    SCCOL_MoreText(
                            scColumn*   col,
                            Bool&       flag );


/*====================== STREAM OPERATIONS ====================*/

// @func Returns an id to the stream a column/flow set contains.
//
status scIMPL_EXPORT    SCCOL_GetStream(
                            scColumn*   column,     // @parm <c scColumn> containing stream.
                            scStream*&  stream );   // @parm <c scStream> pointer to be filled.

// @func Writes stream to file using the call back write routine.
// @xref <f SCCOL_GetStream>

status scIMPL_EXPORT    SCSTR_Write(
                            scStream*   stream,     // @parm <c scStream> to write.
                            APPCtxPtr   ioctxptr,   // @parm <t APPCtxPtr> Abstract file i/o type.
                            IOFuncPtr   ioFuncPtr );// @parm <t IOFuncPtr> write function pointer.


// @func Reads stream from file using the call back read routine.
// Use the appropriate calls for file i/o
// <f SCSET_InitRead>, <f SCOBJ_PtrRestore>, <f SCSET_FiniRead>, <f SCCOL_GetStream>
//
status scIMPL_EXPORT    SCSTR_Read(
                            scStream*&  stream,     // @parm Pointer to <c scStream> to be
                                                    // filled in by Composition Toolbox.
                            scSet*      enumTable,  // @parm Pointer enumeration table.
                            APPCtxPtr   ioctxptr,   // @parm <t APPCtxPtr> Abstract file i/o type.
                            IOFuncPtr   ioFuncPtr );// @parm <t IOFuncPtr> write function pointer.


// @func Deletes a stream. If it is associated with a set of linked columns,
// they become a linked set of empty containers. If no columns are
// attached or no redraw is necessary, the scRedispList will be NULL.
// @xref <f SCCOL_GetStream>
//
status scIMPL_EXPORT    SCSTR_Clear(
                            scStream*       stream,     // @parm <c scStream> to delete.
                            scRedispList*   rInfo );    // @parm <c scRedispList>
                                                        // Redisplay info, arg may be zero.



// @func Cuts a stream from its associated containers.
// @xref <f SCCOL_GetStream>
//
status scIMPL_EXPORT    SCSTR_Cut(
                            scStream*       stream,     // @parm Stream to cut.
                            scRedispList*   rInfo );    // @parm <c scRedispList>
                                                        // Redisplay info, arg may be zero.
```

```
                              APPColumn   appName,    // @parm <t APPColumn>, the name used by client.
                              scColumn*&  newID,      // @parm <c scColumn> name of object allocated b
y toolbox.
                              MicroPoint  width,      // @parm <t MicroPoint> width of new text contai
ner.
                              MicroPoint  depth );    // @parm <t MicroPoint> depth of new text contai
ner.


// @func Deletes a container, removes itself from the flowset with
// the text in this container simply spilling over into
// others in the flow set.
// If it is the only column associated in the flow set
// then the stream is also deleted.
// To delete the whole flow set start deleting from the last column
// so that the first column (and the stream) will be deleted last.
status scIMPL_EXPORT    SCCOL_Delete(
                              scColumn*       col,        // @parm Name of <c scColumn> to delete.
                              scRedispList*   rInfo );    // @parm <c scRedispList>
                                                          // Redisplay info, arg may be zero.

// @func Links the two columns together using the following logic:
// COL2 must represent the first column in a flow set.
// COL1 may be anywhere within a (distinct) flow set,
// COL2 will be chained in after COL1. The ordering/reordering
// of the streams is as follows: <nl>
// 1. if COL1 has text and COL2 has no text, the text flows from COL1 to COL2
// <nl>
// 2. if COL2 has text and COL1 has no text, the text flows from COL1 to COL2
// <nl>
// 3. if both COL1 and COL2 have text, the text from COL2 is appended
// to the text in COL1 - this may may create some confusion
// on the user's part, so use with care. <nl>
// NOTE!!!!!! ANY SELECTION IN EITHER FLOW SET WILL NO LONGER BE VALID!!!!!
// @xref <f SCCOL_Unlink>

status scIMPL_EXPORT    SCCOL_Link(
                              scColumn*       col1,       // @parm Name of first <c scColumn>.
                              scColumn*       col2,       // @parm Name of second <c scColumn>.
                              scRedispList*   rInfo );    // @parm Redisplay info, arg may be zero.


// @func Unlinks a column from its chain. The column passed in becomes
// an empty container, and the stream remains intact. If the column
// is the only column in a chain, it is a no-op.
// <nl>NOTE!!!!!! ANY SELECTION IN THE FLOW SET
// COLUMN WILL NO LONGER BE VALID!!!!!
// @xref <f SCCOL_Link>

status scIMPL_EXPORT    SCCOL_Unlink(
                              scColumn*       col,        // @parm Name of <c scColumn> to unlink.
                              scRedispList*   rInfo );    // @parm <c scRedispList>
                                                          // Redisplay info, arg may be zero.


// @func Severs the link between the two columns. The stream is left in
// the first logical container set. The text is left in a
// uncomposed state.<NL>
// NOTE!!!!!! ANY SELECTION IN THE FLOWSET
// COLUMN WILL NO LONGER BE VALID!!!!!
//
status scIMPL_EXPORT    SCFS_Split(
                              scColumn*   col1,       // @parm <c scColumn> prior to split.
                              scColumn*   col2 );     // @parm <c scColumn> after split.


// @func Resizes a column. If the column has an irregular shape
// (such as a polygon), the width and depth values of the container
// are updated, but no reflowing occurs. The width and depth are independent
// of text flow. Width is always the horizontal dimenision and depth
// the vertical dimension.
//
```

```
/* ================================================================= */
/* ====================  SCTYPESPECLISTT   ========================== */
/* ================================================================= */

// @func Safely allocate.
status scIMPL_EXPORT    SCTSL_Alloc(
                            scTypeSpecList*&  tsl ); // @parm <t scTypeSpecList>
// @func Safely delete.
status scIMPL_EXPORT    SCTSL_Delete(
                            scTypeSpecList*&  tsl ); // @parm <t scTypeSpecList>


/* ================================================================== */
/* ======================  SCREDISPLIST   =========================== */
/* ================================================================== */

// @func Safely allocate.
status scIMPL_EXPORT    SCRDL_Alloc(
                            scRedispList*& rdl );   // @parm <c scRedispList>


// @func Safely delete.
status scIMPL_EXPORT    SCRDL_Delete(
                            scRedispList*& rdl );    // @parm <c scRedispList>


/* ================================================================== */
/* =========================  SCAPPTEXT   =========================== */
/* ================================================================== */


// @func Safely allocate.
status scIMPL_EXPORT    SCAPPTXT_Alloc(
                           stTextImportExport*& atxt );     // @parm <c scAPPText>

// @func Safely delete.
status scIMPL_EXPORT    SCAPPTXT_Delete(
                           stTextImportExport* atxt );      // @parm <c scAPPText>




/* ================================================================== */
/* =====================  SCCHARSPECLIST   ========================== */
/* ================================================================== */

// @func Safely allocate.
status scIMPL_EXPORT    SCCHTS_Alloc(
                            scSpecLocList*& cslist,         // @parm <c scCharSpecList>
                            scStream*           stream );       // @parm <c scStream> to associate w
ith.

// @func Safely delete.
status scIMPL_EXPORT    SCCHTS_Delete(
                            scSpecLocList*& cslist );           // @parm <c scCharSpecList>



/* ================================================================== */
/* ======================== CONTAINER MESSAGES ===================== */
/* ================================================================== */

// @func Creates a new column/container within the Composition Toolbox universe.
// The appName is simply a notational convenience for the client. It is
// presumed that the client is maintaining some sort of container structure
// and the appName is typically pointing to the clients structure. In all
// conversations with the client we use this name. If it is 0 we will simply
// fill in our name for it.
//
status scIMPL_EXPORT    SCCOL_New(
/* ======================
FLOW SET        a set of linked columns which contain a single stream
COLUMN          an area to flow text into, text MAY extend outside the
                column depending upon constraints, another name for a column
                is a TEXT CONTAINER
STREAM          a set of content units/paragraphs
CONTENT UNIT    a paragraph that contains characters and specs associated
                with the characters
SELECTION       a range of text, each flow set may only contain one selection

 * ====================== */


/*====================== SYSTEM LEVEL MESSAGES ===============*/
// @func status | SCENG_Init | This must be called before any other
// calls are made into the Toolbox; it initializes and sets
// toolobx behavior. The base error is the number added to
// <t status> errors when exceptions are re-raised across the api.
//
// @parm int | baseError | Value to add to <t status> values
// if we are re-raising exceptions across the API.
//
status scIMPL_EXPORT    SCENG_Init( int baseError = 0 );



// @func Closes the Composition Toolbox; this releases all memory
// that the Toolbox has allocated. All references into the
// Toolbox become invalid.
//
status scIMPL_EXPORT    SCENG_Fini( void );



// The following three calls are optional. Their use will guarantee that
// the Toolbox can always recover from an out of memory condition,
// given that the application can get it back to a previous state.
//

// Holds memory to guarantee that if recomposition fails, we will
// be able to revert to previous state.
//
// [ ] [ ]
status scIMPL_EXPORT    SCENG_RetainMemory( void );




// Informs Toolbox to use retained memory -
// only for use in a recovery operation.
//
// [ ] [ ]
status scIMPL_EXPORT    SCENG_UseRetainedMemory( void );




// Releases retained memory; recomposition has been successfully completed.
//
// [ ] [ ]
status scIMPL_EXPORT    SCENG_ReleaseMemory( void );


/* ==================================================================== */
/* ============= OBJECT ALLOCATION ==================================== */
/* ==================================================================== */

// If the application wants to allocate its objects it can do
// so bearing in mind that it must have set up an exception
// handler, otherwise the client may use the following to
// allocate and free the objects. Accessing the data within
// these objects should not present exception problems if the
// correct accessor methods are used.
```

```
//<html><pre>
/****************************************************************************

    File:       SCAPPINT.H

    $Header: /Projects/Toolbox/ct/SCAPPINT.H 2     5/30/97 8:45a Wmanis $

    Contains:   The portable c application interface prototypes

                    SCENG_xxxxx     - messages to the text engine

                    SCFS_xxxxxx     - messages to flowsets

                    SCCOL_xxxxx     - messages to columns

                    SCSEL_xxxxx     - messages to a selection

                    SCSTR_xxxxx     - messages to streams

                    SCSCR_xxxxx     - messages to the scrap

                    SCHRGN_xxxx     - messages to regions

                    SCCHTS_xxxx     - messages to scCharSpecList

                    SCTSL_xxxxx     - messages to scTypeSpecList

                    SCAPPTXT_xx     - messages to scAPPText

                    SCRDL_xxxxx     - messages to scRedisplayList

    Written by: Manis

    Copyright (c) 1989-1994 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

@doc

****************************************************************************/

#ifndef _H_SCAPPINT
#define _H_SCAPPINT

#ifdef SCMACINTOSH
#pragma once
#endif

#include "sctypes.h"
#include "sccharex.h"

// [doc] [test]

class stTextImportExport;       // in scapptex.h  ( scTextExchange )

class scTypeSpecList;   // in scpubobj.h
class scLineInfoList;   // in scpubobj.h
class scRedispList;     // in scpubobj.h
class scSpecLocList;    // in scpubobj.h

class scSet;            // see scset.h

class scImmediateRedisp;// in scpubobj.h

class clField;      // in sccallbk.h

// TERMINOLOGY
```

```
        int                     fParaOffset;
        int                     fStartOffset;
        int                     fEndOffset;
};

/* ============================================================
===== */

inline void scAnnotation::Clear()
{
        fAnnotate               = false;
        fParaOffset             = -1;
        fStartOffset    = -1;
        fEndOffset              = -1;
}

/* ============================================================
===== */

inline scAnnotation::scAnnotation( UCS2 *ch, int paraoffset, int s
tart, int end )
{
        Set( ch, paraoffset, start, end );
}

/* ============================================================
===== */

inline scAnnotation::scAnnotation()
{
        Clear();
}

/* ============================================================
===== */

#endif
```

```
/*=================================================================
=

        File:           scannota.h

        $Header: /Projects/Toolbox/ct/SCANNOTA.H 2     5/30/97 8:4
5a Wmanis $

        Contains:       Generalized annotation structure.

        Written by:     Manis

        Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
        All rights reserved.

        This notice is intended as a precaution against inadverten
t publication
        and does not constitute an admission or acknowledgment tha
t publication
        has occurred or constitute a waiver of confidentiality.

        Composition Toolbox software is the proprietary
        and confidential property of Stonehand Inc.

=================================================================*
/

#ifndef _H_SCANNOTA
#define _H_SCANNOTA

#include "sctypes.h"
#include <string.h>

class scAnnotation {

public:
                                scAnnotation( UCS2*, int, int, int
 );
                                scAnnotation();

        void            Set( UCS2 *, int, int, int );
        void            Clear( void );

        UCS2            fCharStr[32];
        Bool            fAnnotate;                      // if true
 apply, else clear
```

```
                    col, col->GetAPPName(),
                    col->Width(), col->Depth(),
                    col->GetVertFlex() ? "VFLEX" : "noflex" );

    if ( contentLevel ) {
        scContUnit* p;

        for ( p = col->GetStream(); p; p = p->GetNext() )
            SCDebugTrace( 0, scString( "\tpara 0x%08x\n" ), p );
    }
}

/* ==================================================================== */

void scIMPL_EXPORT SCDebugParaSpecs( scSelection* sel )
{
#if 1
    scSelection sorted( *sel );

    sorted.Sort();

    scContUnit* cu = sorted.GetMark().fPara;
    for ( ; cu && cu != sorted.GetPoint().fPara->GetNext(); cu = cu->GetNext() )
        cu->DebugParaSpecs();
#else
    if ( sel->IsSliverCursor() )
        sel->GetMark().fPara->DebugParaSpecs();
#endif
}

/* ==================================================================== */

void scIMPL_EXPORT SCSTR_Debug( scStream* str )
{
    str->STRDbgPrintInfo( );
}

/* ==================================================================== */

#endif /* DEBUG */

/* ==================================================================== */
/* ==================================================================== */
/* ==================================================================== */
```

```
    return stat;

}

#endif

/* ================================================================== */
/* ================================================================== */
/* ================================================================== */

void scIMPL_EXPORT      SCCOL_InvertExtents( scColumn*      col,
                                             HiliteFuncPtr  func,
                                             APPDrwCtx      drawCtx )
{
    status stat = scSuccess;
    try {
        col->InvertExtents( func, drawCtx );
    }
    IGNORE_RERAISE;
}

/* ================================================================== */

#if SCDEBUG > 1

long scIMPL_EXPORT      SCCOL_DebugSize( scColumn* col )
{
    return sizeof( scColumn ) + ( col->GetLinecount( ) * sizeof( scTextline ) );
}

/* ================================================================== */

long scIMPL_EXPORT      SCSTR_DebugSize( scStream* stream )
{
    return stream->STRDebugSize(  );
}

/* ================================================================== */

void scIMPL_EXPORT      SCCOL_InvertRegion( scColumn*       col,
                                            HiliteFuncPtr   func,
                                            APPDrwCtx       drawCtx )
{
    status stat = scSuccess;
    try {
        scFlowDir fd( col->GetFlowdir( ) );
        if ( col->GetRgn() )
            RGNInvertSlivers( col->GetRgn(), drawCtx, func, col->GetSize(), fd.IsVertical() );
    }
    IGNORE_RERAISE;
}

/* ================================================================== */

void scIMPL_EXPORT      SCDebugColumnList( void )
{
    scColumn* col;

    SCDebugTrace( 0, scString( "Toolbox Column list start\n" ) );

    for ( col = scColumn::GetBaseContextList( ); col; col = col->GetContext() ) {
        SCDebugTrace( 0, scString( "\tcol 0x%08x appname 0x%08x\n" ), col, col->GetAPPName() );
    }
    SCDebugTrace( 0, scString( "Toolbox Column list end\n" ) );

}

/* ================================================================== */

void scIMPL_EXPORT  SCDebugColumn( scColumn*    col,
                                   int      contentLevel )
{
    SCDebugTrace( 0, scString( "Column 0x%08x appname 0x%08x %d %d %s\n" ),
status scIMPL_EXPORT    SCSEL_GetStream( const scSelection* selection,
                                         scStream*&         stream,
                                         TypeSpec&          ts )

{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSEL_GetStream" ) );

    try {
        stream  = selection->GetStream();
        ts      = selection->GetSpecAtStart();
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSEL_GetStream" ) );
    return scSuccess;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCSTR_NthParaSelect( scStream*      streamID,
                                             long           nthPara,
                                             scSelection*   select )

{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSTR_NthParaSelect" ) );

    try {
        select->NthPara( streamID, nthPara );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSTR_NthParaSelect" ) );
    return stat;
}

/* ==================================================================== */

#ifdef _RUBI_SUPPORT

status scIMPL_EXPORT    SCSEL_GetAnnotation( scSelection*   select,
                                             int            nth,
                                             scAnnotation&  annotation )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSEL_GetAnnotation" ) );

    try {
        select->GetAnnotation( nth, annotation );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSEL_GetAnnotation" ) );
    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCSEL_ApplyAnnotation( scSelection*         select,
                                               const scAnnotation&  annotation,
                                               scRedispList*        redisplayListH )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSEL_ApplyAnnotation" ) );

    try {
        select->ApplyAnnotation( annotation, redisplayListH );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSEL_ApplyAnnotation" ) );
```

```
                                        scRedispList*   damage )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSEL_Iter" ) );

    try {
        select->Iter( func, damage );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSEL_Iter" ) );
    return stat;
}

/* ====================================================================== */

status scIMPL_EXPORT    SCSTR_Iter( scStream*       stream,
                                    SubstituteFunc  func,
                                    scRedispList*   damage )

{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSTR_Iter" ) );

    try {
        stream->Iter( func, damage );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSTR_Iter" ) );
    return stat;
}

/* ====================================================================== */
// deprecated

status scIMPL_EXPORT    SCSTR_Search( scStream*         stream,
                                      const UCS2*       findString,
                                      SubstituteFunc    func,
                                      scRedispList*     damage )

{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSTR_Search" ) );

    try {
        throw( scERRnotImplemented );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSTR_Search" ) );
    return scSuccess;
}

/* ====================================================================== */
// deprecated

status scIMPL_EXPORT    SCSEL_FindString( scSelection*  select,
                                          const UCS2*   findString )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSEL_FindString" ) );

    try {
        throw( scERRnotImplemented );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSEL_FindString" ) );
    return scSuccess;
}

/* ====================================================================== */
```

```
    ExitMonitor( scString( "SCSEL_CopyText" ) );
    return stat;
}


/* ================================================================== */

status scIMPL_EXPORT    SCSEL_PasteText ( scSelection*  selection,
                                          scScrapPtr    scrap,
                                          TypeSpec      style,
                                          scRedispList* redispList )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSEL_PasteText" ) );

    try {
        selection->PasteText( (scStream*)scrap, style, redispList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSEL_PasteText" ) );
    return stat;
}

/* ================================================================== */


status scIMPL_EXPORT    SCSCR_ConToSys ( scScrapPtr             scrap,
                                         SystemMemoryObject&    pSysConBlock )
{
    status      stat = scSuccess;
    scContUnit* para = (scContUnit*)scrap;

    EnterMonitor( scString( "SCSCR_ConToSys" ) );

    try {
        if ( scrap->IsClass( "scColumn" ) )
            para = ((scColumn*)scrap)->GetStream();

        if ( para->IsClass( "scContUnit" ) )
            ((scStream*)para)->STRWriteMemText( false, pSysConBlock );
        else
            raise( scERRidentification );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSCR_ConToSys" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCSCR_SysToCon( scScrapPtr&     scrapP,
                                        const scChar*   sysScrapPtr,
                                        TypeSpec        ts )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSCR_SysToCon" ) );

    try {
        scrapP = scStream::ReadMemText( ts, sysScrapPtr );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSCR_SysToCon" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCSEL_Iter( scSelection*    select,
                                    SubstituteFunc  func,
    ExitMonitor( scString( "SCSEL_SetTextStyle" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCSEL_TextTrans ( scSelection*  select,
                                          eChTranType   trans,
                                          int           numChars,       // this is a modifier for th
e string
                                          scRedispList  *redispList )

{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSEL_TextTrans" ) );

    try {
        select->TextTrans( trans, numChars, redispList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSEL_TextTrans" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCSEL_CutText ( scSelection*    selection,
                                        scScrapPtr&     scrap,
                                        scRedispList*   redispList )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSEL_CutText" ) );

    try {
        selection->CutText( scrap, redispList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSEL_CutText" ) );
    return stat;
}

/*================================================================== */

status scIMPL_EXPORT    SCSEL_ClearText( scSelection*   selection,
                                         scRedispList*  redispList )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSEL_ClearText" ) );

    try {
        selection->ClearText( redispList, true );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSEL_ClearText" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCSEL_CopyText ( scSelection*   selection,
                                         scScrapPtr&    scrap )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSEL_CopyText" ) );

    try {
        selection->CopyText( scrap );
    }
    IGNORE_RERAISE;
    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCSEL_Hilite( scSelection*  select,
                                      HiliteFuncPtr DrawRect )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSEL_Hilite" ) );

    try {
        select->ValidateHilite( DrawRect );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSEL_Hilite" ) );
    return stat;
}

/* ==================================================================== */
/*================== EDITING MESSAGES ==================*/

status scIMPL_EXPORT    SCSEL_InsertKeyRecords( scSelection*    select,
                                                short           keyCount,
                                                scKeyRecord*    keyRecords, /* array of key recds */
                                                scRedispList*   redispList )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSEL_InsertKeyRecords" ) );

    try {
        select->KeyArray( keyCount, keyRecords, redispList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSEL_InsertKeyRecords" ) );
    return stat;
}

/* ==================================================================== */

status SCSEL_InsertField( scSelection*      sel,
                          const clField&    field,
                          TypeSpec&         spec,
                          scRedispList*     redisplist )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSEL_InsertAnnotation" ) );

    try {
        sel->InsertField( field, spec, redisplist );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSEL_InsertAnnotation" ) );
    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCSEL_SetTextStyle ( scSelection*   selection,
                                             TypeSpec       style,
                                             scRedispList*  redispList )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSEL_SetTextStyle" ) );

    try {
        selection->SetStyle( style, redispList );
    }
    IGNORE_RERAISE;
    EnterMonitor( scString( "SCSEL_Restore" ) );

    try {
        scColumn* col = scColumn::FindFlowset( stream );

            // we cannot create a selection if there is no layout
        raise_if( col == 0, scERRstructure );

        select = col->FlowsetGetSelection();

        select->Restore( &mark, &point, stream, geometryChange );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSEL_Restore" ) );
    return stat;
}


/* ==================================================================== */

status scIMPL_EXPORT    SCCOL_SelectSpecial( scColumn*          col,
                                             const scMuPoint&   pt,
                                             eSelectModifier    selectMod,
                                             scSelection*&      selectID )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCCOL_SelectSpecial" ) );

    try {
        col->SelectSpecial( pt, selectMod, selectID );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_SelectSpecial" ) );
    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCSEL_Move( scSelection*    select,
                                    eSelectMove     moveSelect )
{
    status stat = scSuccess;

    EnterMonitor( scString( "SCSEL_Move" ) );

    try {
        select->MoveSelect( moveSelect );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSEL_Move" ) );

    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCSEL_Extend( scSelection*  select,
                                      eSelectMove   moveSelect )
{
    status stat = scSuccess;

    EnterMonitor( scString( "SCSEL_Move" ) );

    try {
        select->Extend( moveSelect );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSEL_Move" ) );
    try {
        col->InitialSelection( typespec, selectID );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_InitialSelect" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCSEL_Decompose ( scSelection*      select,
                                          scStreamLocation& mark,
                                          scStreamLocation& point )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSEL_Decompose" ) );

    try {
        select->Decompose( mark, point );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSEL_Decompose" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCSEL_Decompose2( scSelection*      select,
                                          scStreamLocation& mark,
                                          scStreamLocation& point )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSEL_Decompose" ) );

    try {
        select->Decompose2( mark, point );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSEL_Decompose" ) );
    return stat;
}

/* ================================================================== */
status scIMPL_EXPORT    SCSEL_Invalidate( scSelection*& select )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSEL_Invalidate" ) );

    try {
        if ( select )
            select->Invalidate();
    }
    IGNORE_RERAISE;

    select = 0;

    ExitMonitor( scString( "SCSEL_Invalidate" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCSEL_Restore( const scStream*          stream,
                                       const scStreamLocation&  mark,
                                       const scStreamLocation&  point,
                                       scSelection*&            select,
                                       Bool                     geometryChange )
{
    status stat = scSuccess;
```

```
{
    status      stat = scSuccess;
    EnterMonitor( scString( "SCCOL_StartSelect" ) );

#if SCDEBUG > 1
    SCDebugTrace( 2, scString( "SCCOLStartSelect %d %d\n" ), pt.x, pt.y );
#endif

    try {
        col->StartClick( pt, DrawRect, mat, selectID );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_StartSelect" ) );
    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCCOL_StartSelect( scColumn*             col,
                                           scStreamLocation&     mark,
                                           const scMuPoint&      pt,
                                           HiliteFuncPtr         DrawRect,
                                           APPDrwCtx             mat,
                                           scSelection*&         selectID )
{
    status      stat = scSuccess;

    EnterMonitor( scString( "SCCOL_StartSelect" ) );

    try {
        col->StartShiftClick( mark, pt, DrawRect, mat, selectID );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_StartSelect" ) );
    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCCOL_ExtendSelect( scColumn*            col,
                                            const scMuPoint&     pt,
                                            HiliteFuncPtr        DrawRect,
                                            APPDrwCtx,
                                            scSelection*         select )
{
    status      stat = scSuccess;

    EnterMonitor( scString( "SCCOL_ExtendSelect" ) );

//  SCDebugTrace( 0, scString( "SCCOLExtendSelect ENTER %d %d\n" ), pt.x, pt.y );

    try {
        raise_if( select == 0, scERRinput );
        col->ContinueClick( pt, DrawRect, select );
    }
    IGNORE_RERAISE;

//  SCDebugTrace( 0, scString( "SCCOLExtendSelect EXIT %d %d\n" ), pt.x, pt.y );

    ExitMonitor( scString( "SCCOL_ExtendSelect" ) );
    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCCOL_InitialSelect ( scColumn*     col,
                                              TypeSpec&     typespec,
                                              scSelection*& selectID )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCCOL_InitialSelect" ) );
    ExitMonitor( scString( "SCSEL_GetAPPText" ) );
    return stat;
}

/* ================================================================= */

status scIMPL_EXPORT    SCCOL_Write( scColumn*  col,
                                     APPCtxPtr  ctxPtr,
                                     IOFuncPtr  writeFunc )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCCOL_Write" ) );

    try {
        col->StartWrite( ctxPtr, writeFunc );
        scTBObj::WriteNullObject( ctxPtr, writeFunc );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_Write" ) );
    return stat;
}

/* ================================================================= */

status scIMPL_EXPORT    SCCOL_Read( APPColumn       appName,
                                    scColumn*&      col,
                                    scSet*          enumTable,
                                    APPCtxPtr       ctxPtr,
                                    IOFuncPtr       readFunc )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCCOL_Read" ) );

    try {
        col = (scColumn*)scTBObj::StartRead( enumTable, ctxPtr, readFunc );
        scAssert( scTBObj::StartRead( enumTable, ctxPtr, readFunc ) == 0 );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_Read" ) );
    return stat;
}

/* ================================================================= */
/*================== SELECTION MESSAGES ==================*/

status scIMPL_EXPORT    SCCOL_ClickEvaluate ( scColumn*         col,
                                              const scMuPoint&  pt,
                                              REAL&             dist )
{
    status      stat = scSuccess;
    scMuPoint   cmpt = pt;

    EnterMonitor( scString( "SCCOL_ClickEvaluate" ) );

    try {
        col->ClickEvaluate( cmpt, dist );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_ClickEvaluate" ) );
    return stat;
}

/* ================================================================= */

status scIMPL_EXPORT    SCCOL_StartSelect( scColumn*        col,
                                           const scMuPoint& pt,
                                           HiliteFuncPtr DrawRect,
                                           APPDrwCtx        mat,
                                           scSelection*&    selectID )
    ExitMonitor( scString( "SCAPPTXT_Delete" ) );

    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCFS_PasteAPPText( scColumn*            col,
                                           stTextImportExport&      appText,
                                           scRedispList*    redispList )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCFS_PasteAPPText" ) );

    try {
        col->PasteAPPText( appText, redispList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCFS_PasteAPPText" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCSEL_PasteAPPText( scSelection*            sel,
                                            stTextImportExport&     appText,
                                            scRedispList*    redispList )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSEL_PasteAPPText" ) );

    try {
        sel->PasteAPPText( appText, redispList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSEL_PasteAPPText" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCSTR_GetAPPText( scStream*     stream,
                                          stTextImportExport&   appText )
{
    status stat = scSuccess;
    /*CLIPSTUFF*/
    EnterMonitor( scString( "SCSTR_GetAPPText" ) );

    try {
        stream->CopyAPPText ( appText );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSTR_GetAPPText" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCSEL_GetAPPText( scSelection*         selection,
                                          stTextImportExport&   appText )
{
    status stat = scSuccess;
    /*CLIPSTUFF*/
    EnterMonitor( scString( "SCSEL_GetAPPText" ) );

    try {
        selection->CopyAPPText ( appText );
    }
    IGNORE_RERAISE;
```

```
/* ==================================================================== */

status scIMPL_EXPORT    SCWriteTextFile ( scStream* stream,
                                          APPCtxPtr ctxPtr,
                                          IOFuncPtr writeFunc )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCWriteTextFile" ) );

    try {
        stream->STRWriteTextFile( ctxPtr, writeFunc, false );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCWriteTextFile" ) );
    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCTextFileToScrap ( scScrapPtr& scrapH,
                                            APPCtxPtr   ctxPtr,
                                            IOFuncPtr   readFunc )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCTextFileToScrap" ) );

    try {
        TypeSpec nullSpec;

        scStream* stream = scStream::ReadTextFile( nullSpec, ctxPtr, readFunc, 0 );
        scrapH  = stream;
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCTextFileToScrap" ) );
    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCAPPTXT_Alloc( stTextImportExport*& apptext )
{
    status stat = scSuccess;

    EnterMonitor( scString( "SCAPPTXT_Alloc" ) );

    apptext = 0;

    try {
        apptext = &stTextImportExport::MakeTextImportExport( 1 );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCAPPTXT_Alloc" ) );

    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCAPPTXT_Delete( stTextImportExport* apptext )
{
    status stat = scSuccess;

    EnterMonitor( scString( "SCAPPTXT_Delete" ) );

    try {
        apptext->release();
    }
    IGNORE_RERAISE;
```

```
/* ================================================================== */

status scIMPL_EXPORT    SCCOL_Unlink ( scColumn*       col,
                                       scRedispList*   redispList )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCCOL_Unlink" ) );

    try {
        col->Unlink( redispList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_Unlink" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCFS_Split( scColumn*   col1,
                                    scColumn*   col2 )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCFS_Split" ) );

    try {
        col1->BreakChain( col2 );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCFS_Split" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCCOL_GetStream ( scColumn*     col,
                                          scStream*&    stream )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCCOL_GetStream" ) );

    try {
        stream = col->GetStream ();
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_GetStream" ) );
    return stat;
}

/* ================================================================== */


status scIMPL_EXPORT    SCFS_ReadTextFile ( scColumn*       col,
                                            TypeSpec        spec,
                                            APPCtxPtr       ctxPtr,
                                            IOFuncPtr       readFunc,
                                            scRedispList*   redispList )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCFS_ReadTextFile" ) );

    try {
        col->ReadTextFile( spec, ctxPtr, readFunc, redispList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCFS_ReadTextFile" ) );
    return stat;
}
```

```
status scIMPL_EXPORT    SCCOL_QueryMargins( scColumn*   col,
                                            scXRect&    xrect )
{
    status  stat = scSuccess;

    EnterMonitor( scString( "SCCOL_QueryMargins" ) );

    try {
        col->QueryMargins( xrect );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_QueryMargins" ) );
    return stat;
}

/* ================================================================ */

status scIMPL_EXPORT    SCCOL_Size( scColumn*   col,
                                    scSize&     size )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCCOL_Size" ) );

    try {
        col->QuerySize( size );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_Size" ) );
    return stat;
}

/* ================================================================ */

status scIMPL_EXPORT    SCOBJ_Enumerate( scTBObj*   obj,
                                         long&      objEnumerate )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCOBJ_Enumerate" ) );

    try {
        if ( obj->IsClass( "scColumn" ) )
            ((scColumn*)obj)->Enumerate( objEnumerate );
        else if ( obj->IsClass( "scContUnit" ) )
            ((scStream*)obj)->DeepEnumerate( objEnumerate );
        else
            raise( scERRstructure );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCOBJ_Enumerate" ) );
    return stat;
}

/* ================================================================ */

status scIMPL_EXPORT    SCCOL_Link( scColumn*       col1,
                                    scColumn*       col2,
                                    scRedispList*   redispList )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCCOL_Link" ) );

    try {
        col1->Link( col2, true, redispList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_Link" ) );
    return stat;
}
```

```
    try {
        delete enumTable, enumTable = 0;
        scColumn::Update( redispList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSET_FiniRead" ) );

    return stat;
}

/* ====================================================================== */

status scIMPL_EXPORT    SCSET_Abort( scSet*& enumTable )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSET_Abortt" ) );
    try {
        enumTable->DeleteAll();
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSET_Abortt" ) );
    return stat;
}

/* ====================================================================== */

status scIMPL_EXPORT    SCOBJ_PtrRestore( scTBObj*  obj,
                                          scSet*    enumTable )
{
    status stat = scSuccess;

    EnterMonitor( scString( "SCOBJ_PtrRestore" ) );

    try {
        long    i,
                limit = enumTable->GetNumItems();

        for ( i = 0; i < limit; i++ ) {
            scTBObj* ptr = (scTBObj*)enumTable->Get( i );
            if ( ptr )
                ptr->RestorePointers( enumTable );
        }
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCOBJ_PtrRestore" ) );

    return stat;
}

/* ====================================================================== */

status scIMPL_EXPORT    SCCOL_QueryInkExtents( scColumn*    col,
                                               scXRect& xrect )
{
    status  stat = scSuccess;

    EnterMonitor( scString( "SCCOL_QueryInkExtents" ) );

    try {
        col->ComputeInkExtents();
        xrect = col->GetInkExtents();
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_QueryInkExtents" ) );
    return stat;
}

/* ====================================================================== */
```

```
    try {
        scAssert( str == cslist.GetStream() );
        str->SetParaSpecList( cslist, rInfo );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSTR_PARATSListSet" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCExternalSize ( scColumn*  col,
                                         long&      size )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCExternalSize" ) );

    try {
        col->ExternalSize( size );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCExternalSize" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCTB_ZeroEnumeration( void )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCTB_ZeroEnumeration" ) );

    try {
        scColumn*   col = scColumn::GetBaseContextList();
        for ( ; col; col = col->GetContext() )
            col->ZeroEnumeration();
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCTB_ZeroEnumeration" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCSET_InitRead( scSet*& enumTable,
                                        long    maxsize )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSET_InitRead" ) );

    try {
        enumTable = SCNEW scSet;
        enumTable->SetNumSlots( maxsize );
        enumTable->SetRetainMem( true );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSET_InitRead" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCSET_FiniRead( scSet*          enumTable,
                                        scRedispList*   redispList )
{
    status stat = scSuccess;

    EnterMonitor( scString( "SCSET_FiniRead" ) );
    EnterMonitor( scString( "SCSEL_PARATSList" ) );

    try {
        sel->GetParaSpecList( cslist );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSEL_PARATSList" ) );
    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCSEL_PARATSList( scSelection*      sel,
                                          scTypeSpecList&   tsList )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSEL_PARATSList" ) );

    try {
        sel->GetParaSpecList( tsList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSEL_PARATSList" ) );
    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCSTR_PARATSList( scStream*         stream,
                                          scSpecLocList&    cslist )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSTR_PARATSList" ) );

    try {
        stream->GetParaSpecList( cslist );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSTR_PARATSList" ) );
    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCSTR_CHTSListSet ( scStream*               str,
                                            const scSpecLocList&    csList,
                                            scRedispList*           redispList )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSTR_CHTSListSet" ) );

    try {
        scAssert( str == csList.GetStream() );
        str->SetCharSpecList( csList, redispList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSTR_CHTSListSet" ) );
    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCSTR_PARATSListSet( scStream*              str,
                                             const scSpecLocList&   cslist,
                                             scRedispList*          rInfo )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSTR_PARATSListSet" ) );
```

```
status scIMPL_EXPORT    SCCOL_FlowJustify( scColumn*        col,
                                           eVertJust    attributes )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCCOL_FlowJustify" ) );

    try {
        col->SetVJ( attributes );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_FlowJustify" ) );
    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCSTR_ChCount( scStream*    stream,
                                       long&        count )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSTR_ChCount" ) );

    try {
        stream->ChCount( count );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSTR_ChCount" ) );
    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCSEL_TSList ( scSelection*     selection,
                                       scTypeSpecList&  tsList )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSEL_TSList" ) );

    try {
        selection->GetTSList( tsList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSEL_TSList" ) );
    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCSEL_CHTSList( scSelection*        selection,
                                        scSpecLocList&  csList )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSEL_CHTSList" ) );

    try {
        selection->GetCharSpecList( csList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSEL_CHTSList" ) );
    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCSEL_PARATSList( scSelection*      sel,
                                          scSpecLocList&   cslist )
{
    status stat = scSuccess;
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCSTR_TSList ( scStream*        stream,
                                       scTypeSpecList&  tsList )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSTR_TSList" ) );

    try {
        stream->STRGetTSList( tsList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSTR_TSList" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCSTR_ParaTSList ( scStream*        stream,
                                       scTypeSpecList&  tsList )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSTR_ParaTSList" ) );

    try {
        stream->GetParaTSList( tsList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSTR_ParaTSList" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCSTR_CHTSList ( scStream*      stream,
                                         scSpecLocList& csList )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSTR_CHTSList" ) );

    try {
        stream->GetCharSpecList( csList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSTR_CHTSList" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCCOL_SetDepthNVJ( scColumn*           col,
                                           MicroPoint       depth,
                                           scRedispList     *redispList )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCCOL_SetDepthNVJ" ) );

    try {
        col->SetDepthNVJ ( depth, redispList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_SetDepthNVJ" ) );
    return stat;
}

/* ================================================================== */
/* ================================================================== */

status scIMPL_EXPORT    SCHRGN_RectIn( const HRgnHandle hrgH,
                                       const scXRect&   xrect )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCHRGN_RectIn" ) );

    try {
        stat = RectInHRgn( hrgH, xrect ) ? scSuccess : scNoAction;
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCHRGN_RectIn" ) );
    return stat;
}


/* ================================================================== */

#endif


/* ================================================================== */

status scIMPL_EXPORT    SCCOL_Update( scColumn*         col,
                                      const scXRect&    xrect,
                                      APPDrwCtx         mat )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCCOL_Update" ) );

    try {
        col->Draw ( xrect, mat );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_Update" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCCOL_UpdateLine( scColumn*           col,
                                          scImmediateRedisp&   lineDamage,
                                          APPDrwCtx           mat )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCCOL_UpdateLine" ) );

    try {
        col->UpdateLine( lineDamage, mat );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_UpdateLine" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCCOL_TSList ( scColumn*          col,
                                       scTypeSpecList&  tsList )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCCOL_TSList" ) );

    try {
        col->GetTSList( tsList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_TSList" ) );
```

```
/* ================================================================== */

status scIMPL_EXPORT    SCHRGN_Translate( HRgnHandle    hrgH,
                                          MicroPoint    x,
                                          MicroPoint    y )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCHRGN_Translate" ) );

    try {
        TranslateHRgn( hrgH, x, y );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCHRGN_Translate" ) );
    return stat;
}


/* ================================================================== */

status scIMPL_EXPORT    SCHRGN_Inset( HRgnHandle    hrgH,
                                      MicroPoint    x,
                                      MicroPoint    y )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCHRGN_Inset" ) );

    try {
        InsetHRgn( hrgH, x, y, true );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCHRGN_Inset" ) );
    return stat;
}


/* ================================================================== */

status scIMPL_EXPORT    SCHRGN_SetEmpty( HRgnHandle hrgH )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCHRGN_SetEmpty" ) );

    try {
        SetEmptyHRgn( hrgH );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCHRGN_SetEmpty" ) );
    return stat;
}


/* ================================================================== */

status scIMPL_EXPORT    SCHRGN_SliverSize( HRgnHandle      hrgH,
                                           MicroPoint&  sliverSize )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCHRGN_SliverSize" ) );

    try {
        sliverSize = RGNSliverSize( hrgH );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCHRGN_SliverSize" ) );
    return stat;
}
```

```
                                    const HRgnHandle   b,
                                    HRgnHandle         dstRgnH )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCHRGN_Sect" ) );

    try {
        SectHRgn( a, b, dstRgnH );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCHRGN_Sect" ) );
    return stat;
}


/* ====================================================================== */

status scIMPL_EXPORT    SCHRGN_Union( const HRgnHandle  a,
                                      const HRgnHandle  b,
                                      HRgnHandle        dstRgnH )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCHRGN_Union" ) );

    try {
        UnionHRgn( a, b, dstRgnH );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCHRGN_Union" ) );
    return stat;
}


/* ====================================================================== */

status scIMPL_EXPORT    SCHRGN_Diff( const HRgnHandle   a,
                                     const HRgnHandle   b,
                                     HRgnHandle         dstRgnH )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCHRGN_Diff" ) );

    try {
        DiffHRgn( a, b, dstRgnH );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCHRGN_Diff" ) );
    return stat;
}


/* ====================================================================== */

status scIMPL_EXPORT    SCHRGN_Xor( const HRgnHandle    a,
                                    const HRgnHandle    b,
                                    HRgnHandle          dstRgnH )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCHRGN_Xor" ) );

    try {
        XorHRgn( a, b, dstRgnH );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCHRGN_Xor" ) );
    return stat;
}
```

```
status scIMPL_EXPORT    SCHRGN_PtIn( const HRgnHandle    hrgH,
                                     const scMuPoint&    pt )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCHRGN_PtIn" ) );

    try {
        stat = PtInHRgn( hrgH, pt ) ? scSuccess : scNoAction;
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCHRGN_PtIn" ) );
    return stat;
}


/* ==================================================================== */

status scIMPL_EXPORT    SCHRGN_Rect( HRgnHandle     hrgH,
                                     const scXRect& xrect )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCHRGN_Rect" ) );

    try {
        RectHRgn( hrgH, xrect );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCHRGN_Rect" ) );
    return stat;
}


/* ==================================================================== */

status scIMPL_EXPORT    SCHRGN_Poly( HRgnHandle         hrgH,
                                     const scVertex*    verts )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCHRGN_Poly" ) );

    try {
        PolyHRgn( hrgH, verts );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCHRGN_Poly" ) );
    return stat;
}


/* ==================================================================== */

status scIMPL_EXPORT    SCHRGN_Copy( HRgnHandle         dstRgn,
                                     const HRgnHandle   srcRgn )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCHRGN_Copy" ) );

    try {
        CopyHRgn( dstRgn, srcRgn );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCHRGN_Copy" ) );
    return stat;
}


/* ==================================================================== */

status scIMPL_EXPORT    SCHRGN_Sect( const HRgnHandle   a,
/* ================================================================ */

status scIMPL_EXPORT    SCHRGN_New( HRgnHandle& hrgH,
                                    MicroPoint  sliverSize )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCHRGN_New" ) );

    try {
        hrgH = NewHRgn( sliverSize );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCHRGN_New" ) );
    return stat;
}


/* ================================================================ */

status scIMPL_EXPORT    SCHRGN_Dispose( HRgnHandle hrgH )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCHRGN_Dispose" ) );

    try {
        DisposeHRgn( hrgH );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCHRGN_Dispose" ) );
    return stat;
}


/* ================================================================ */

status scIMPL_EXPORT    SCHRGN_Empty( HRgnHandle hrgH )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCHRGN_Empty" ) );

    try {
        stat = EmptyHRgn( hrgH ) ? scSuccess : scNoAction;
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCHRGN_Empty" ) );
    return stat;
}


/* ================================================================ */

status scIMPL_EXPORT    SCHRGN_Equal( const HRgnHandle  a,
                                      const HRgnHandle b )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCHRGN_Equal" ) );

    try {
        stat = EqualHRgn( a, b ) ? scSuccess : scNoAction;
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCHRGN_Equal" ) );
    return stat;
}


/* ================================================================ */
```

```
    ExitMonitor( scString( "SCCOL_ClearPoly" ) );
    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCCOL_CopyPoly ( scColumn*  col,
                                         scVertex*& vert )
{
    status stat = scSuccess;
    /*CLIPSTUFF*/
    EnterMonitor( scString( "SCCOL_CopyPoly" ) );

    try {
//      col->CopyPoly( vert );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_CopyPoly" ) );
    return stat;
}

/*=============================== REGIONS ==========================*/

status scIMPL_EXPORT    SCCOL_PasteRgn ( scColumn*          col,
                                         const HRgnHandle   rgnH,
                                         scRedispList*      redispList )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCCOL_PasteRgn" ) );

    try {
        col->PasteRgn( rgnH, redispList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_PasteRgn" ) );
    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCCOL_ClearRgn ( scColumn*      col,
                                         scRedispList   *redispList )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCCOL_ClearRgn" ) );

    try {
        col->ClearShape( redispList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_ClearRgn" ) );
    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCCOL_CopyRgn ( scColumn*   col,
                                        HRgnHandle& rgnH )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCCOL_CopyRgn" ) );

    try {
        col->CopyRgn( rgnH );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_CopyRgn" ) );
    return stat;
}
```

```
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_GetHorzFlex" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCCOL_GetFlowDirection( scColumn*   col,
                                                scFlowDir&  flodir )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCCOL_GetFlowDirection" ) );

    try {
        flodir = col->GetFlowdir();
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_GetFlowDirection" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCCOL_SetFlowDirection( scColumn*           col,
                                                const scFlowDir&    flodir )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCCOL_SetFlowDirection" ) );

    try {
        col->FlowsetSetFlowdir( flodir );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_SetFlowDirection" ) );
    return stat;
}

/* ================================================================== */

#if defined( scColumnShape )
status scIMPL_EXPORT    SCCOL_PastePoly ( scColumn*         col,
                                          const scVertex*   vert,
                                          scRedispList*     redispList )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCCOL_PastePoly" ) );

    try {
//      col->PastePoly ( vert, redispList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_PastePoly" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCCOL_ClearPoly ( scColumn*     col,
                                          scRedispList* redispList )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCCOL_ClearPoly" ) );

    try {
        col->ClearShape( redispList );
    }
    IGNORE_RERAISE;
    status stat = scSuccess;
    EnterMonitor( scString( "SCCOL_ClearVertFlex" ) );

    try {
        col->SetVertFlex ( false, redispList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_ClearVertFlex" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCCOL_SetHorzFlex ( scColumn*       col,
                                            scRedispList*   redispList )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCCOL_SetHorzFlex" ) );

    try {
        col->SetHorzFlex ( true, redispList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_SetHorzFlex" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCCOL_ClearHorzFlex ( scColumn*     col,
                                              scRedispList* redispList )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCCOL_ClearHorzFlex" ) );

    try {
        col->SetHorzFlex ( false, redispList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_ClearHorzFlex" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCCOL_GetVertFlex( scColumn*    col,
                                           Bool&        tf )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCCOL_GetVertFlex" ) );

    try {
        tf = col->GetVertFlex( );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_GetVertFlex" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCCOL_GetHorzFlex ( scColumn*   col,
                                            Bool&       tf )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCCOL_GetHorzFlex" ) );

    try {
        tf = col->GetHorzFlex( );
    try {
        stat = col->HasText( ) ? scSuccess : scNoAction;
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_HasText" ) );
    return stat;
}

/* ================================================================== */

// tests to see if there is more text than is in this column
// this would set the flag to true if:
//          there is text in subsequent linked columns
//          there is unformatted text that will not fit in this column

status scIMPL_EXPORT    SCCOL_MoreText( scColumn*   col,
                                        Bool&       flag )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCCOL_MoreText" ) );

    try {
        flag = col->MoreText( );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_MoreText" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCCOL_LinePositions ( scColumn*         col,
                                              scLineInfoList*   lineInfo,
                                              long&             nLines,
                                              Bool&             moreText )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCCOL_LinePositions" ) );

    try {
        col->LineInfo( lineInfo, nLines, moreText );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_LinePositions" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCCOL_SetVertFlex ( scColumn*       col,
                                            scRedispList*   redispList )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCCOL_SetVertFlex" ) );

    try {
        col->SetVertFlex ( true, redispList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_SetVertFlex" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCCOL_ClearVertFlex ( scColumn*         col,
                                              scRedispList* redispList )
{
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCSTR_Split( scStream*      stream1,
                                     scContUnit*    cu,
                                     scStream*&     stream2 )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSTR_Split" ) );

    try {
        stream2 = stream1->Split( cu );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSTR_Split" ) );
    return stat;
}

/* ================================================================== */
/* compare streams for equality, this tests content and specs
 * scSuccess == equality
 */

status scIMPL_EXPORT    SCSTR_Compare( const scStream* str1,
                                       const scStream* str2 )
{
    status stat = scSuccess;

    EnterMonitor( scString( "SCSTR_Compare" ) );

    try {
        stat = str1->Compare( str2 ) ? scSuccess : scNoAction;
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSTR_Compare" ) );

    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCCOL_SetSize( scColumn*        col,
                                       MicroPoint       width,
                                       MicroPoint       depth,
                                       scRedispList*    redispList )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCCOL_SetSize" ) );


    try {
        if  ( width < 0 || depth < 0 )
            raise( scERRinput );

        col->Resize( width, depth, redispList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_SetSize" ) );
    return stat;
}

/* ================================================================== */
// is there any text associated with this column

status scIMPL_EXPORT    SCCOL_HasText( scColumn* col )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCCOL_HasText" ) );
```

```
}

/* ==================================================================== */

status scIMPL_EXPORT    SCFS_PasteStream ( scColumn*           col,
                                           scStream*        streamID,
                                           scRedispList*    redispList )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCFS_PasteStream" ) );

    try {
        col->FlowsetPasteStream( streamID, redispList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCFS_PasteStream" ) );
    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCSTR_Clear ( scStream*     stream,
                                      scRedispList* redispList )
{
    status      stat = scSuccess;
    scColumn*   col;
    scTextline* txl;

    EnterMonitor( scString( "SCSTR_Clear" ) );

    try {
        if ( stream ) {
            txl = stream->GetFirstline();
            if  ( txl ) {
                col = txl->GetColumn( );
                if  ( col )
                    col->FlowsetClearStream ( redispList );
                else
                    raise( scERRstructure );
            }
            else if ( stream->FindColumn( col ) )
                col->FlowsetClearStream( redispList );
            else
                    /* if no layout structure associated with stream */
                stream->STRFree();
        }
        else
            raise( scNoAction );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSTR_Clear" ) );
    return stat;
}

/* ==================================================================== */
// Extracts a scContUnit from a scStreamLocation for use with SCSTR_Split

status scIMPL_EXPORT    SCSEL_GetContUnit( scContUnit*& mark,
                                           scContUnit*& point,

                                           const scSelection* sl )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSTR_GetContUnit" ) );

    try {
        sl->GetContUnits( mark, point );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSTR_GetContUnit" ) );
    try {
        stream = scStream::STRFromFile( enumTable, ctxPtr, readFunc );
        scAssert( scTBObj::StartRead( enumTable, ctxPtr, readFunc ) == 0 );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSTR_Read" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCSTR_Write( scStream*  stream,
                                     APPCtxPtr  ctxPtr,
                                     IOFuncPtr  writeFunc )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSTR_Write" ) );

    try {
        stream->STRWriteToFile( ctxPtr, writeFunc );
        scTBObj::WriteNullObject( ctxPtr, writeFunc );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSTR_Write" ) );
    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCSTR_Cut ( scStream*   streamID,
                                    scRedispList*   redispList )
{
    status      stat = scSuccess;
    scColumn*   col;
    scTextline* txl;

    EnterMonitor( scString( "SCSTR_Cut" ) );

    try {
        txl = streamID->GetFirstline();
        if  ( txl )
            col =  txl->GetColumn( );
        else
            col = scColumn::FindFlowset( streamID );

        if ( col )
            col->FlowsetCutStream ( streamID, redispList );
        else
            raise( scERRstructure );
    }
    IGNORE_RERAISE;

    return stat;
}

/* ================================================================== */

status scIMPL_EXPORT    SCSTR_Copy( const scStream* stream,
                                    scStream*&      newStream )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSTR_Copy" ) );

    try {
        stream->STRCopy( newStream );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSTR_Copy" ) );
    return stat;
```

```
status scIMPL_EXPORT    SCSCR_TSList( scScrapPtr        scrap,
                                      scTypeSpecList&   tsList )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSCR_TSList" ) );

    try {
        if ( scrap->IsClass( "scColumn" ) )
            ((scColumn *)scrap)->GetTSList( tsList );
        else if ( scrap->IsClass( "scContUnit" ) )
            ((scStream*)scrap)->GetTSList ( tsList );
        else
            stat = scERRidentification;
    }
    IGNORE_RERAISE;

    ExitMonitor( scString(  "SCSCR_TSList" ) );
    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCSCR_Free( scScrapPtr scrap )
{
    status  stat = scSuccess;
    long    bytesFreed;

    EnterMonitor( scString( "SCSCR_Free" ) );

    try {
        if ( !scrap )
            ;
        else if ( scrap->IsClass( "scColumn" ) )
            ((scColumn*)scrap)->FreeScrap();
        else if ( scrap->IsClass( "scContUnit" ) )
            ((scContUnit*)scrap)->FreeScrap( bytesFreed );
        else
            raise( scERRidentification );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSCR_Free" ) );
    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCCOL_Delete( scColumn*     col,
                                      scRedispList* redispList )

{
    status stat = scSuccess;

    EnterMonitor( scString( "SCCOL_Delete" ) );

    try {
        col->Delete( redispList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_Delete" ) );
    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCSTR_Read( scStream*&  stream,
                                    scSet*      enumTable,
                                    APPCtxPtr   ctxPtr,
                                    IOFuncPtr   readFunc )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSTR_Read" ) );
```

```
                                        IOFuncPtr      writeFunc )
{
    status  stat = scSuccess;
    EnterMonitor( scString( "SCSCR_Write" ) );

    try {
        if ( scrap->IsClass( "scColumn" ) ) {
            scColumn* col = (scColumn*)scrap;
            col->ZeroEnumeration();
            col->StartWrite( ctxPtr, writeFunc );
        }
        else if ( scrap->IsClass( "scStream" ) ) {
            scStream* stream = (scStream*)scrap;
            stream->STRZeroEnumeration();
            stream->STRWriteToFile( ctxPtr, writeFunc );
        }
        else
            raise( scERRidentification );
        scTBObj::WriteNullObject( ctxPtr, writeFunc );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSCR_Write" ) );
    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCSCR_ReadCol( scScrapPtr&  scrap,
                                       scSet*       enumTable,
                                       APPCtxPtr    ctxPtr,
                                       IOFuncPtr    readFunc )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSCR_ReadCol" ) );

    try {
        scColumn* col;
        col = (scColumn*)scTBObj::StartRead( enumTable, ctxPtr, readFunc );
        scAssert( scTBObj::StartRead( enumTable, ctxPtr, readFunc ) == 0 );
        scrap = col;
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCSCR_ReadCol" ) );
    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCSCR_ReadStream( scScrapPtr&   scrapH,
                                          scSet*        enumTable,
                                          APPCtxPtr     ctxPtr,
                                          IOFuncPtr     readFunc )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCSCR_ReadStream" ) );

    scrapH = 0;

    try {
        scStream* stream = scStream::STRFromFile( enumTable, ctxPtr, readFunc );
        scAssert( scTBObj::StartRead( enumTable, ctxPtr, readFunc ) == 0 );
        scrapH  = stream;
    }
    IGNORE_RERAISE;

    ExitMonitor( scString(  "SCSCR_ReadStream" ) );
    return stat;
}

/* ==================================================================== */
```

```
    EnterMonitor( scString( "SCFS_GetRecompose" ) );

    try {
        tf = col->GetRecomposition();
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCFS_GetRecompose" ) );
    return stat;
}

/* ====================================================================== */

status scIMPL_EXPORT    SCFS_Recompose( scColumn*           col,
                                        scRedispList*   redispList )
{
    status stat = scSuccess;

    EnterMonitor( scString( "SCFS_Recompose" ) );

    try {
        col->RecomposeFlowset( LONG_MAX, redispList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCFS_Recompose" ) );
    return stat;
}

/* ====================================================================== */

status scIMPL_EXPORT    SCFS_Recompose( scColumn*           col,
                                        long            ticks,
                                        scRedispList*   redispList )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCFS_Recompose" ) );

    try {
        col->RecomposeFlowset( ticks, redispList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCFS_Recompose" ) );
    return stat;
}

/* ====================================================================== */


status scIMPL_EXPORT    SCCOL_New( APPColumn    appName,
                                   scColumn*&   col,
                                   MicroPoint   width,
                                   MicroPoint   depth )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCCOL_New" ) );

    try {
        raise_if( width < 0 || depth < 0, scERRinput );
        col = SCNEW scColumn( appName, width, depth );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_New" ) );
    return stat;
}

/* ====================================================================== */

status scIMPL_EXPORT    SCSCR_Write( scScrapPtr     scrap,
                                     APPCtxPtr      ctxPtr,
    try {
        delete rdlist, rdlist = 0;
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCRDL_Delete" ) );

    return stat;
}

/* ====================================================================== */
/* Recompose a single column with extreme prejudice */

status scIMPL_EXPORT    SCCOL_Recompose( scColumn*          col,
                                         scRedispList*  redispList )
{
    status stat = scSuccess;

    EnterMonitor( scString( "SCCOL_Recompose" ) );

    try {
        col->Rebreak( redispList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCOL_Recompose" ) );
    return stat;
}

/* ====================================================================== */
/* Recompose a single column with extreme prejudice */

status scIMPL_EXPORT    SCRebreakCol ( scColumn*        col,
                                       scRedispList*    redispList )
{
    status stat = scSuccess;

    EnterMonitor( scString( "SCRebreakCol" ) );

    try {
        col->Rebreak2( redispList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCRebreakCol" ) );
    return stat;
}

/* ====================================================================== */

status scIMPL_EXPORT    SCFS_SetRecompose( scColumn* col, Bool tf )
{
    status stat = scSuccess;

    EnterMonitor( scString( "SCFS_SetRecompose" ) );

    try {
        if ( col )
            col->SetRecomposition( tf );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCFS_SetRecompose" ) );
    return stat;
}

/* ====================================================================== */

status scIMPL_EXPORT    SCFS_GetRecompose( scColumn*    col,
                                           Bool&    tf )
{
    status stat = scSuccess;
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCHTS_Delete" ) );

    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCTSL_Alloc( scTypeSpecList*& tslist )
{
    status stat = scSuccess;

    EnterMonitor( scString( "SCTSL_Alloc" ) );

    tslist = 0;

    try {
        tslist = SCNEW scTypeSpecList;
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCTSL_Alloc" ) );

    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCTSL_Delete( scTypeSpecList*& tslist )
{
    status stat = scSuccess;

    EnterMonitor( scString( "SCTSL_Delete" ) );

    try {
        delete tslist, tslist = 0;
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCTSL_Delete" ) );

    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCRDL_Alloc( scRedispList*& rdlist )
{
    status stat = scSuccess;

    EnterMonitor( scString( "SCRDL_Alloc" ) );

    rdlist = 0;

    try {
        rdlist = SCNEW scRedispList;
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCRDL_Alloc" ) );

    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCRDL_Delete( scRedispList*& rdlist )
{
    status stat = scSuccess;

    EnterMonitor( scString( "SCRDL_Delete" ) );
```

```
{
    //  MEMSetRestrictions( memRetain );
    return scSuccess;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCENG_UseRetainedMemory ( void )
{
    //  MEMSetRestrictions( memUseRetained );
    return scSuccess;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCENG_ReleaseMemory ( void )
{
    //  MEMSetRestrictions( memNoRestrictions );
    return scSuccess;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCENG_ChangedTS ( TypeSpec      ts,
                                          eSpecTask     task,
                                          scRedispList* redispList )
{
    status stat = scSuccess;
    EnterMonitor( scString( "SCENG_ChangedTS" ) );

    scCachedStyle::StyleInvalidateCache ( ts );

    try {
        if ( task & eSCDoAll )
            scColumn::ChangedTS( ts, task, redispList );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCENG_ChangedTS" ) );
    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCCHTS_Alloc( scSpecLocList*&   cslist,
                                      scStream*         stream )

    status stat = scSuccess;

    EnterMonitor( scString( "SCCHTS_Alloc" ) );

    cslist = 0;

    try {
        cslist = SCNEW scSpecLocList( stream );
    }
    IGNORE_RERAISE;

    ExitMonitor( scString( "SCCHTS_Alloc" ) );

    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCCHTS_Delete( scSpecLocList*& cslist )
{
    status stat = scSuccess;

    EnterMonitor( scString( "SCCHTS_Delete" ) );

    try {
        delete cslist, cslist = 0;
    #define EnterMonitor( x )
    #define ExitMonitor( x )
#endif



/* ==================================================================== */

void        BRKInitMach( void );
void        BRKFreeMach( void );


char* stoneVersion =  __DATE__" - "__TIME__;

status scIMPL_EXPORT    SCENG_Init( int baseError )
{
        // The following are pool definitions that are passed
        // to the initiailization of the memory manager.
        // The last pool is the default memory allocation pool
        // all others are fixed size pools, these are not sorted
        // at this time
    static scPoolInfo   objPools[] = {
        { sizeof( scTextline ),             0 },
        { sizeof( scContUnit ),             0 },
        { sizeof( scAbstractArray ),        0 },
        { 0,                                0 }
    };

#ifndef useCPLUSEXCEPTIONS
        // if we are not using C++ exceptions - initialize our own
        // exception manager.
    scExceptContext::Initialize( 0 );
#endif

    status stat = scSuccess;

    gBaseError  = baseError;

    try {
        MEMInit( objPools );                // initialize memory manager
//      scAssert( sizeof( CharRecord ) == ( sizeof( long ) * 2 ) );
        BRKInitMach();                      // initialize breaking machine
        scCachedStyle::BuildCache( 16 );    // build internal spec cache
        gInited                 = true;
    }
    IGNORE_RERAISE;

    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCENG_Fini( void )
{
    status stat = scSuccess;

    try {
        scAssert( gInited );
        BRKFreeMach( );
        scColumn::FiniCTXList();
        scCachedStyle::DeleteCache( );

        gInited = false;
        MEMFini();
    }
    IGNORE_RERAISE;

    return stat;
}

/* ==================================================================== */

status scIMPL_EXPORT    SCENG_RetainMemory ( void )
```

```
/***************************************************************************

    File:       SCAPI.C


    $Header: /Projects/Toolbox/ct/SCAPI.CPP 3     5/30/97 8:45a Wmanis $

    Contains:   Application Program Interface for the
                Stonehand Composition Toolbox. For the most part
                this file is simply a bottle nect module. All
                documentation for the functions contained within
                are found in scappint.h.

    Written by: Manis


    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.


***************************************************************************/


#include "scappint.h"
#include "scpubobj.h"


#include "scannota.h"
#include "scapptex.h"
#include "sccolumn.h"
#include "scexcept.h"
#include "scstcach.h"
#include "scglobda.h"
#include "scmem.h"
#include "scparagr.h"
#include "scregion.h"
#include "scset.h"
#include "scstream.h"
#include "sctextli.h"


static int      gInited;
static int      gBaseError;
int             scDebugTrace = 0;

/* ================================================================== */


#if 0
    static int gInputLevel;

        // if scDebugTrace is set to a value >0 all calls into
        // the toolbox will be traced, may be useful for understanding
        // a behavior or pointing the finger!

    inline void EnterMonitor( const scChar *str )
    {
        scAssert( gInited );
        SCDebugTrace( 0, scString( "\n+%s\n" ), str );
    }

    inline void ExitMonitor( const scChar *str )
    {
        SCDebugTrace( 0, scString( "-%s\n" ), str );
    }
#else
```

```
    SCDebugStr( buf );
}

/* ======================================================================= */

void SCDebugBreak( void )
{
    DebugStr( "\pSCDEBUGBREAK" );
}

/* ======================================================================= */
```

```
/*************************************************************
    File:       SC_UTLTC.C

    $Header: /Projects/Toolbox/ct/SC_UTMAC.CPP 2     5/30/97 8:45a Wmanis $

    Contains:   Think C untilities

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

*************************************************************/

//#include "scport.h"
//#include "capplica.h"
//#include "constant.h"
//#include "tbutilities.h"

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <Dialogs.h>
#include <SegLoad.h>
#include <QuickDraw.h>
#include <OSUtils.h>

#include "scTypes.h"

#if defined( THINK_CPLUS ) && THINK_CPLUS < 0x0700
    #include <pascal.h>
#else
    #include <Strings.h>
#endif

Boolean gSCUseSysBreak;

/* ======================================================================== */

void SCDebugStr( const char *str )
{
    char buf[256];

    strcpy( buf, str );

//  if ( gApplication->TestDebuggerPresence() ) {
        if (gSCUseSysBreak)
            SysBreakStr( c2pstr( buf ) );
        else
            DebugStr( c2pstr( buf ) );
//  }
//  else {
//      ParamText( c2pstr( buf ), (StringPtr)"", (StringPtr)"", (StringPtr)"");
//      PositionDialog('ALRT', ALRTgeneral);
//      InitCursor();
//      Alert(ALRTgeneral, NULL);
//  }
}

/* ======================================================================== */

void SCAssertFailed( const scChar *str, const scChar *file, int line )
{
    char buf[256];

    sprintf( buf, "ASSERT FAILED \"%s\" File %s Line %ld", str, file, line );
#endif
}

/*==============================================================*/

void SCSysBeep( long duration )
{
#ifdef _WIN32
        Beep( 500, duration );
#else
        MessageBeep( -1 );
#endif
}

/*==============================================================*/
```

```
/***************************************************************

    File:       SC_UTLWI.C

    $Header: /Projects/Toolbox/ct/SC_UTLWI.CPP 2     5/30/97 8:45a Wmanis $

    Contains:   WINDOWS versions of low level debugging stuff

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.


***************************************************************/

#include "sctypes.h"
#include "scexcept.h"

/*============================================================*/

void SCDebugStr ( const scChar* cstr )
{
    OutputDebugString( cstr );
}

/*============================================================*/

void SCAssertFailed ( const scChar* assertStr,
                      const scChar* file,
                      int           lineNum )
{
    scChar buf[256];

    if ( scStrlen( assertStr ) + scStrlen( file ) + 4 < 256 )
        wsprintf( buf, scString( "ASSERT FAILED \"%s\" file \"%s\" line #%d\n" ),
                  assertStr, file, lineNum );
    else
        scStrcpy( buf, scString( "ASSERT STRING TOO LONG!!!!\n" ) );

    SCDebugStr( buf );

#if SCDEBUG < 1
    raise( scERRassertFailed );
#else
    SCDebugBreak();

        // set doit to true if you want to raise an exception
    int doit = 0;
    if ( doit )
        raise( scERRassertFailed );
#endif
}

/*============================================================*/

void SCDebugBreak( void )
{
#if SCDEBUG > 1
    DebugBreak();
#else
    #ifdef _WIN32
            Beep( 500, 100 );
    #else
            MessageBeep( -1 );
    #endif
```

```
status SystemMemoryObject::SetHandleSize( long newsize )
{
#if defined( SCWINDOWS )
    fSYSHandle = GlobalReAlloc( fSYSHandle, newsize, GMEM_MOVEABLE | GMEM_ZEROINIT );
    return fSYSHandle != 0 ? scSuccess : scERRmem;
#elif defined( SCMACINTOSH )
    ::SetHandleSize( fSYSHandle, newsize );
    return MemError() == noErr ? scSuccess : scERRmem;
#endif
}

/* ==================================================================== */

void  *SystemMemoryObject::LockHandle( void )
{
#if defined( SCWINDOWS )
    return GlobalLock( fSYSHandle );
#elif defined( SCMACINTOSH )
    HLock( fSYSHandle );
    return *fSYSHandle;
#endif
}

/* ==================================================================== */

void SystemMemoryObject::UnlockHandle( void )
{
#if defined( SCWINDOWS )
    GlobalUnlock( fSYSHandle );
#elif defined( SCMACINTOSH )
    HUnlock( fSYSHandle );
#endif
}

/*====================================================================*/
```

```
/*****************************************************************

    File:       SC_SYSCO.C

    $Header: /Projects/Toolbox/ct/SC_SYSCO.CPP 2     5/30/97 8:45a Wmanis $

    Contains:   Implementation of transfer of clipboard data
                to external format.

    Written by: Lucas

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

*****************************************************************/

/* THESE ARE STUBS AND ARE BY NO MEANS COMPLETE OR ROBUST */

#include "sctypes.h"
#ifdef SCMACINTOSH
#include <Memory.h>
#endif

/* ================================================================== */

SystemMemoryObject::SystemMemoryObject()
{
#if defined( SCWINDOWS )
    fSYSHandle = GlobalAlloc( 0, GPTR ); // since GHND allows only 64k bytes
#elif defined( SCMACINTOSH )
    fSYSHandle = NewHandle( 0 );
#endif
}

/* ================================================================== */

SystemMemoryObject::~SystemMemoryObject()
{
#if defined( SCWINDOWS )
    if ( fSYSHandle )
        GlobalFree( fSYSHandle );
#elif defined( SCMACINTOSH )
    if ( fSYSHandle )
        DisposHandle( fSYSHandle );
#endif
}

/* ================================================================== */

void SystemMemoryObject::ReleaseMem()
{
    fSYSHandle = 0;
}

/* ================================================================== */

long SystemMemoryObject::HandleSize( void )
{
#if defined( SCWINDOWS )
    return (size_t)GlobalSize( fSYSHandle );
#elif defined( SCMACINTOSH )
    return GetHandleSize( fSYSHandle );
#endif
}

/* ================================================================== */
```

```
        case scTabFillAlign:

        case scMinMeasure:
        case scRunAroundBorder:
        case scFirstLine:

            return eSCRebreak;
    }
}

/***************************************************************************/
```

09629370.073400

```
        case scOptLsp:
        case scMaxLsp:
        case scMinWsp:
        case scOptWsp:
        case scMaxWsp:
        case scHyphenation:
        case scHyphLines:
        case scHyphExcep:
        case scHyphMinSize:
        case scPreHyphs:
        case scPostHyphs:
        case scHyphPropensity:
        case scHyphCaps:
        case scHyphAcros:
        case scHyphExtra1:
        case scHyphExtra2:
        default:
            return eSCRetabulate;

        case scColor:
        case scRenderAttribute:
//      case scULShow:
//      case scULpos:
//      case scULthick:
//          return eSCRepaint;

        case scLead:
        case scBaseline:
        case scAboveLead:
        case scBelowLead:
        case scIndLines:
        case scIndAmount:
        case scIndDepth:
        case scIndLeftBL:
        case scIndRightBL:
        case scIndentExtra1:
        case scIndentExtra2:
        case scColNoBreak:
        case scKeepNext:
        case scLinesBefore:
        case scLinesAfter:
        case scWidowOrphanExtra1:
        case scWidowOrphanExtra2:
        case scRag:
        case scForceJust:
        case scRagPattern:
        case scRagZone:
        case scKernMargins:
        case scHLeft:
        case scHRight:
        case scHLeftAmount:
        case scHRightAmount:
        case scRagExtra1:
        case scRagExtra2:
        case scHPuncExtra1:
        case scHPuncExtra2:
        case scDCShow:
        case scDCptSize:
        case scDCsetSize:
        case scDChOffset:
        case scDCvOffset:
        case scDChBorder:
        case scDCvBorder:
        case scDCfont:
        case scDCcolor:
        case scMaxFillChars:
        case scFillPos:
        case scFillChar:
        case scFillAlign:
        case scMaxTabs:
        case scTabPos:
        case scTabAlign:
        case scTabChar:
```

```
/***************************************************************

    File:       SC-SpecChng.c

    $Header: /Projects/Toolbox/ct/SC_SPCHG.CPP 2     5/30/97 8:45a Wmanis $

    Contains:

    When type specs change their are certain types of things that need
    to be done to bring the world back into equilibrium. These tasks
    typically involve REFORMATTING and REPAINT. Since a certain number
    of the formating computations are held with the characters themselves
    the reformatting requires two operations. We will call these
    RETABULATION - correcting the escapement stored with the characters -
    and the LINEBREAKING - the act of breaking text into lines.

    Therefore when a spec changes one or more tasks may need to
    be performed, we want to determine the minimum set of tasks to
    perform to return the world to equilibrium.

    The tasks are performed in the following order:

        TABULATION
        LINE BREAKING
        PAINTING

    Here a few examples of spec changes and what they should cause:

        color change                 - scREPAINT
        word space change            - scREBREAK & scREPAINT
        lead change                  - scREBREAK & scREPAINT
        font change                  - scRETABULATE, scREBREAK & scREPAINT
        pointsize change             - scRETABULATE, scREBREAK & scREPAINT
        setsize change               - scRETABULATE, scREBREAK & scREPAINT
        pair/track kerning change    - scRETABULATE, scREBREAK & scREPAINT

        hyphenation language change - scRETABULATE, scREBREAK & scREPAINT
        # of consecutive hyph change- scREBREAK & scREPAINT


    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.


 ***********************************************************************/

#include "sccallbk.h"

eSpecTask   SpecTaskCalculate( eSpecChange changeType )
{
    switch ( changeType ) {
        case scHoblique:
        case scVoblique:
            return (eSpecTask)((int)eSCRetabulate | (int)eSCRepaint);

        case scLanguage:
        case scFont:
        case scCharTransform:
        case scPointSize:
        case scSetSize:
        case scRotation:
        case scKern:
        case scTrack:
        case scMinLsp:
```

```
/* mapping on reading input buffer or file */

#ifndef noCMinputMap

UCS2 CMinputMap( ushort ch )
{
    return ch;
}

#endif /* noCMinputMap */

/****************************************************************************/

#ifndef noCMmakeKeyRecordTwo

void CMmakeKeyRecordTwo( scKeyRecord&        keyRecord,
                         UCS2                keyCode,
                         GlyphSize           val,
                         TypeSpec            spec,
                         Bool                restoreSelection,
                         scStreamLocation&   mark )
{
    keyRecord.keycode()         = keyCode;
    keyRecord.replacedchar()    = 0;
    keyRecord.escapement()      = val;
    keyRecord.spec()            = spec;
    keyRecord.noop()            = false;
    keyRecord.restoreselect()   = restoreSelection;
    keyRecord.mark()            = mark;
}

#endif /* noCMmakeKeyRecordTwo */

/****************************************************************************/
```

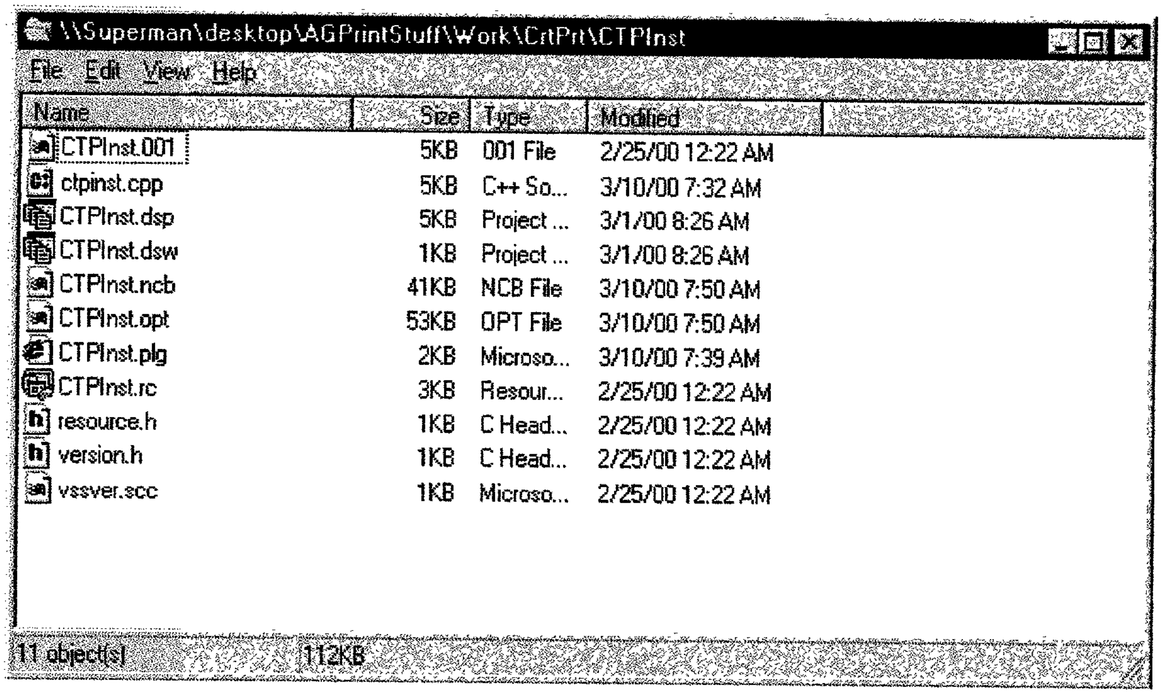
```
            return ch;
        case scNoBreakHyph:
            return '-';

#if 1
        case scTabSpace:
            return 0;
        case scParaEnd:
            return 0;
        case scEndStream:
            return 0;
        case scHardReturn:
            return 0;
#else
        case scTabSpace:
            return 0x00bb;
        case scParaEnd:
            return 0x00b6;
        case scEndStream:
            return 0x00a5;
        case scHardReturn:
            return 'H';
#endif

        case scEmSpace:
        case scEnSpace:
        case scFigureSpace:
        case scThinSpace:
        case scFixRelSpace:
        case scFixAbsSpace:
        case scFillSpace:
        case scVertTab:
        case scNoBreakSpace:
        case scQuadCenter:
        case scQuadLeft:
        case scQuadRight:
        case scQuadJustify:
        case scEmptySpace:
            return ' ';
    }
}

#endif /* noCMctToAPP */

/***************************************************************************/
/* defines whether keyboard input changes model & selection or selection
* only. Called from within Composition Toolboxt prior to keyboard input
* to determine what is about to happen
*/

#ifndef noCMcontent

int CMcontent( UCS2 ch )
{
    switch ( ch ) {
        case scBackSpace:
        case scForwardDelete:
            return -1;

        default:
            return 1;

        case scLeftArrow:
        case scRightArrow:
        case scUpArrow:
        case scDownArrow:
            return false;
    }
}

#endif /* noCMcontent */

/***************************************************************************/
```

```
/*****************************************************************

    File:       SC-CharMap.c

    $Header: /Projects/Toolbox/ct/SC_CHMAP.CPP 6     5/30/97 8:45a Wmanis $

    Contains:   Character mapping between client and toolbox.

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

*****************************************************************/

/*********************************************************************

    to turn off any fo the functions in thismodule define one or
    more of the following values int SCAPPTypes.h, they will turn
    off the appropriate function.

    noCMctToAPP
    noCMappToCT
    noCMmakeKeyRecordTwo
    noCMcontent

**********************************************************************/

#include "sccharex.h"

#define CTL( ch )   ( (ch) - '@' )

/*************************************************************************/
/* this provides character mapping between Mac-keyboard/application
 * to the Composition Toolboxt
 */
#ifndef noCMappToCT

UCS2 CMappToCT( UCS2    ch )
{
    switch ( ch ) {
        case 0x0D:  return scParaSplit;             /* mac enter */
        case 0x08:  return scBackSpace;             /* mac delete */
        case 0x09:  return scTabSpace;              /* mac tab */
        case 0x0a:  return scHardReturn;                /* mac return */

        default:    return (UCS2)ch;
    }
}

#endif /* noCMappToCT */

/*************************************************************************/
/* provides Compositon Toolboxt to application mapping,
 * used pre AppDrawString for rationalize character mapping,
 * used to control characters passed thru, typically used
 * to control things like show invisibles, may be used for
 * other types of character conversion depending on output device
 */

#ifndef noCMctToAPP

UCS2 CMctToAPP( UCS2 ch )
{
    switch ( ch ) {
        default:
              {
                  return p_ == 0;
              }
    int operator!() const
              {
                return p_ == 0;
              }
    int operator==( const ConstRefCountPtr<T>& p ) const
              {
                  return p_ == p.p_;
              }
    int operator!=( const ConstRefCountPtr<T>& p ) const
              {
                  return p_ != p.p_;
              }
    int operator==( const T* p ) const
              {
                  return p_ == p;
              }
    int operator!=( const T* p ) const
              {
                  return p_ != p;
              }

protected:
    T*  p_;
};


template<class T>
class RefCountPtr : public ConstRefCountPtr<T> {
public:
      RefCountPtr( T* ptr = 0 ) :
        ConstRefCountPtr<T>( ptr ){ }

      ~RefCountPtr() { }
  T*    ptr() const
        {
          return p_;
        }
  T*    operator->() const
        {
          return p_;
        }
  T&    operator*() const
        {
          return *p_;
        }
  void  exch(RefCountPtr<T> &p)
        {
          T* tmp = p.p_;
          p.p_ = p_;
          p_ = tmp;
        }
};

#endif
```

```
////////////////////////////////////////////////////////////////
// The following are classes to maintain safe reference count on
// classes derived from RefCount

// T must have RefCount as a public base class
// T may be an incomplete type

template<class T>
class ConstRefCountPtr {
public:
        ConstRefCountPtr() : p_(0) { }
        ConstRefCountPtr( T* p ) : p_( p )
            {
                if ( p_ )
                    p_->incref();
            }

        ~ConstRefCountPtr()
            {
                clear();
            }
        ConstRefCountPtr( const ConstRefCountPtr<T>& p ) :
              p_(p.p_)
            {
              if (p.p_)
                  p.p_->incref();
            }
        ConstRefCountPtr<T> &operator=( const ConstRefCountPtr<T> & p )
            {
              if ( this != &p ) {
                  if (p.p_)
                      p.p_->incref();

                  clear();

                  p_ = p.p_;
              }
              return *this;
            }

    void    moveTo( ConstRefCountPtr<T> &dst )
              {
                  if ( this == &dst )
                      return;

                  dst.clear();

                  dst.p_ = p_;
                  p_ = 0;
              }

    void    clear()
              {
                  if ( p_ && p_->decref())
                      delete p_;
                  p_ = 0;
              }

    const T*  ptr() const
              {
                  return p_;
              }
    const T*  operator->() const
              {
                  return p_;
              }
    const T&  operator*() const
              {
                  return *p_;
              }
    int isNull() const
```

```
//
// Copyright (c) 1996, Stonehand Inc.  All rights reserved.
//

#ifndef _H_REFCNT
#define _H_REFCNT

  // a base class for reference counting

#ifdef _DEBUG
    void SCDebugBreak( void );
#endif

class RefCount {
public:
            RefCount() : refcnt_(0)
            {
#ifdef _DEBUG
                magic_ = 0xbabaabab;
#endif
            }
            RefCount(const RefCount &) : refcnt_(0)
            {
#ifdef _DEBUG
                magic_ = 0xbabaabab;
#endif
            }

  virtual   ~RefCount()
              {
                  if ( refcnt_ )
                      throw( -1 );
              }

  int       decref()      // return 1 if it should be deleted
              {
#ifdef _DEBUG
                  static void* test = 0;
                  if ( this == test )
                      SCDebugBreak();
#endif
                  return --refcnt_ <= 0;
              }

  void      incref()
              {
#ifdef _DEBUG
                  static void* test = 0;
                  if ( this == test )
                      SCDebugBreak();
#endif
                  ++refcnt_;
              }

  int       refcnt()
              {
                  return refcnt_;
              }
#ifdef _DEBUG
  unsigned  magic()
              {
                return magic_;
              }
#endif

private:
#ifdef _DEBUG
  unsigned  magic_;
#endif
  int       refcnt_;
};
```

Untitled

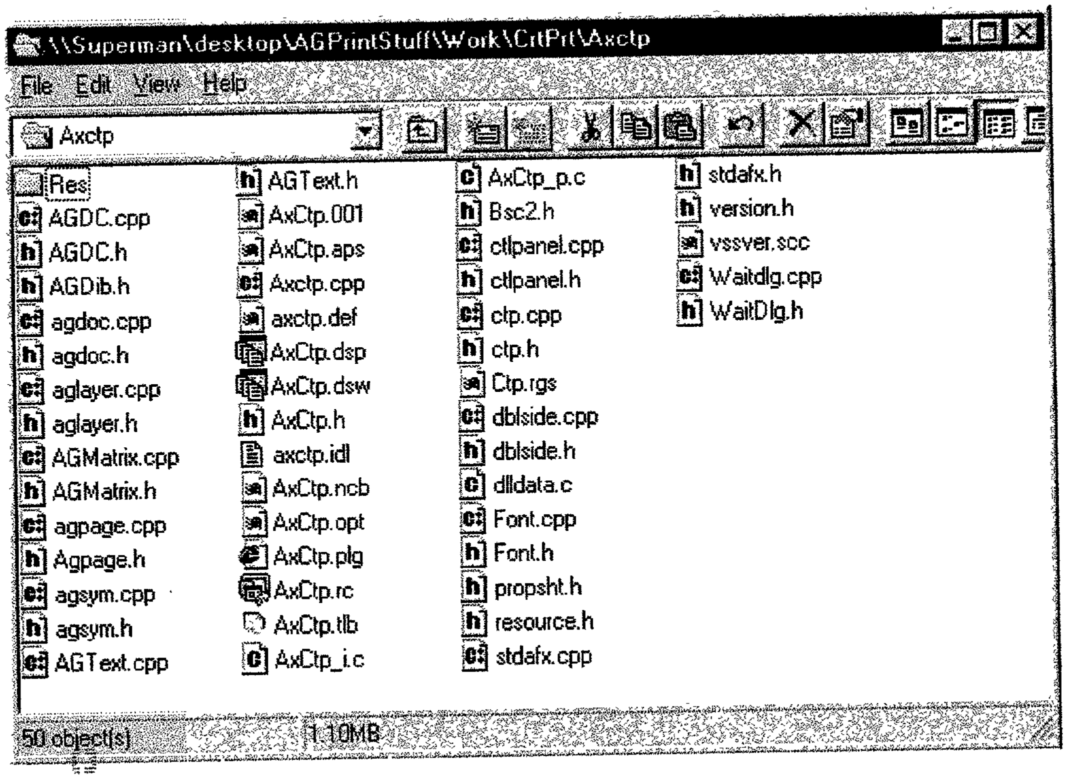

\\Superman\desktop\AGPrintStuff\Work\CrtPrt\CTPInst

Name	Size	Type	Modified
CTPInst.001	5KB	001 File	2/25/00 12:22 AM
ctpinst.cpp	5KB	C++ So...	3/10/00 7:32 AM
CTPInst.dsp	5KB	Project ...	3/1/00 8:26 AM
CTPInst.dsw	1KB	Project ...	3/1/00 8:26 AM
CTPInst.ncb	41KB	NCB File	3/10/00 7:50 AM
CTPInst.opt	53KB	OPT File	3/10/00 7:50 AM
CTPInst.plg	2KB	Microso...	3/10/00 7:39 AM
CTPInst.rc	3KB	Resour...	2/25/00 12:22 AM
resource.h	1KB	C Head...	2/25/00 12:22 AM
version.h	1KB	C Head...	2/25/00 12:22 AM
vssver.scc	1KB	Microso...	2/25/00 12:22 AM

11 object(s) 112KB

[illegible]

Untitled

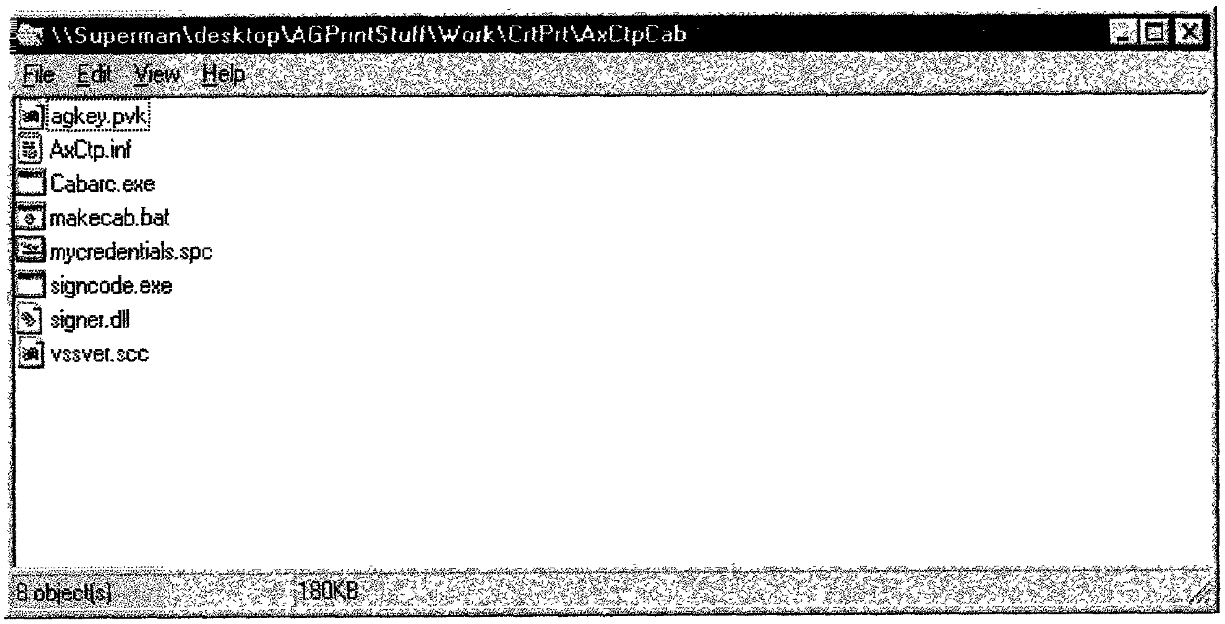

\\Superman\desktop\AGPrintStuff\Work\CrtPrt\Axctp

```
Res            AGText.h       AxCtp_p.c      stdafx.h
AGDC.cpp       AxCtp.001      Bsc2.h         version.h
AGDC.h         AxCtp.aps      ctlpanel.cpp   vssver.scc
AGDib.h        Axctp.cpp      ctlpanel.h     Waitdlg.cpp
agdoc.cpp      axctp.def      ctp.cpp        WaitDlg.h
agdoc.h        AxCtp.dsp      ctp.h
aglayer.cpp    AxCtp.dsw      Ctp.rgs
aglayer.h      AxCtp.h        dblside.cpp
AGMatrix.cpp   axctp.idl      dblside.h
AGMatrix.h     AxCtp.ncb      dlldata.c
agpage.cpp     AxCtp.opt      Font.cpp
Agpage.h       AxCtp.plg      Font.h
agsym.cpp      AxCtp.rc       propsht.h
agsym.h        AxCtp.tlb      resource.h
AGText.cpp     AxCtp_i.c      stdafx.cpp
```

50 object(s)   1.10MB

[illegible]

Intitled



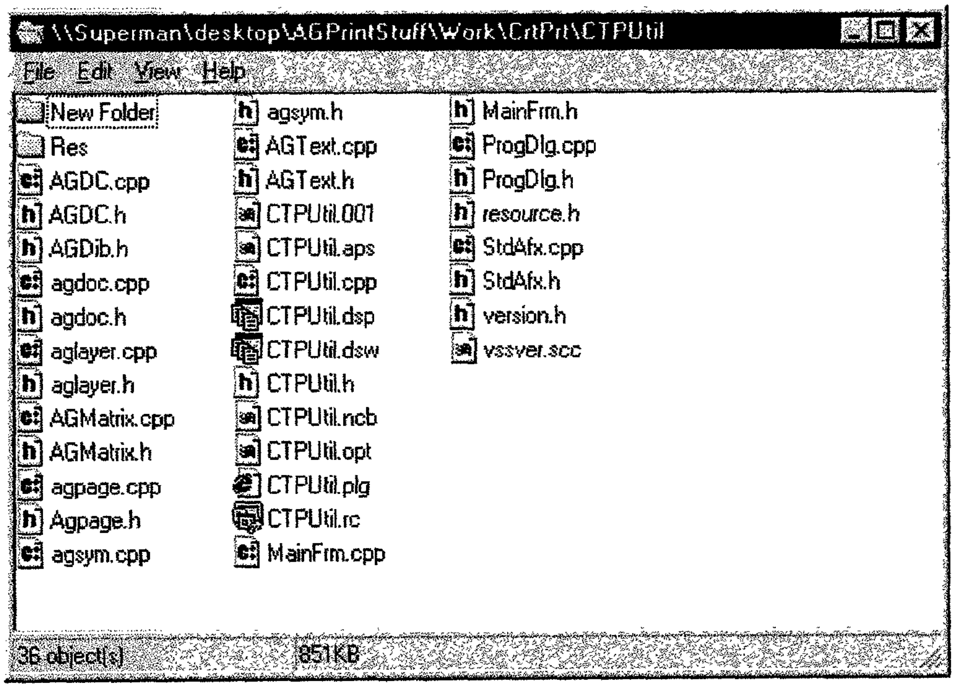
\\Superman\desktop\AGPrintStuff\Work\CrtPrt\AxCtpCab

agkey.pvk
AxCtp.inf
Cabarc.exe
makecab.bat
mycredentials.spc
signcode.exe
signer.dll
vssver.scc

8 object(s) 180KB

09620370.072100

Jntitled

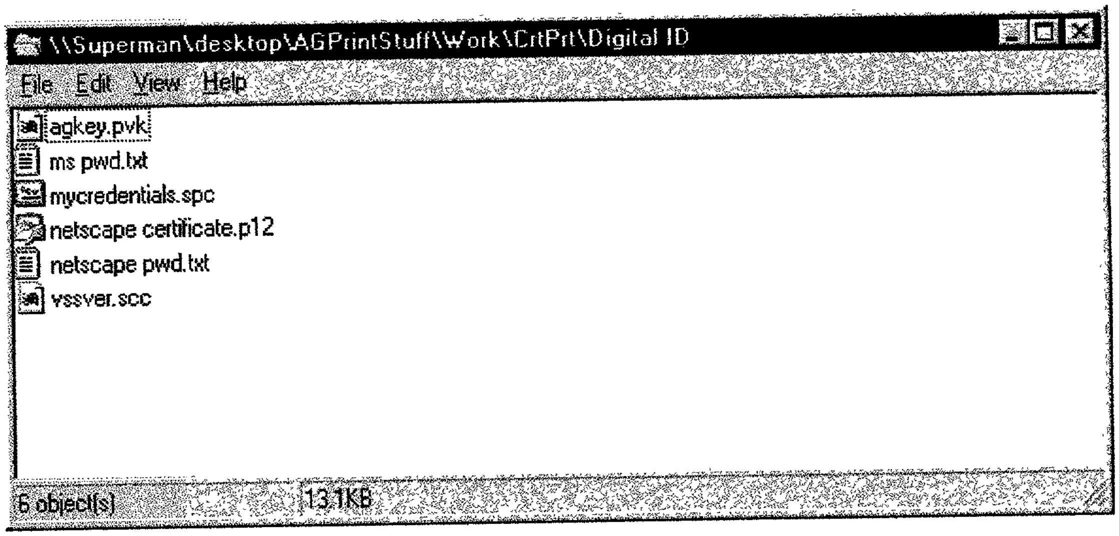


\\Superman\desktop\AGPrintStuff\Work\CrtPrt\CTPUtil

```
New Folder      agsym.h         MainFrm.h
Res             AGText.cpp      ProgDlg.cpp
AGDC.cpp        AGText.h        ProgDlg.h
AGDC.h          CTPUtil.001     resource.h
AGDib.h         CTPUtil.aps     StdAfx.cpp
agdoc.cpp       CTPUtil.cpp     StdAfx.h
agdoc.h         CTPUtil.dsp     version.h
aglayer.cpp     CTPUtil.dsw     vssver.scc
aglayer.h       CTPUtil.h
AGMatrix.cpp    CTPUtil.ncb
AGMatrix.h      CTPUtil.opt
agpage.cpp      CTPUtil.plg
Agpage.h        CTPUtil.rc
agsym.cpp       MainFrm.cpp
```

36 object(s)   851KB

Untitled

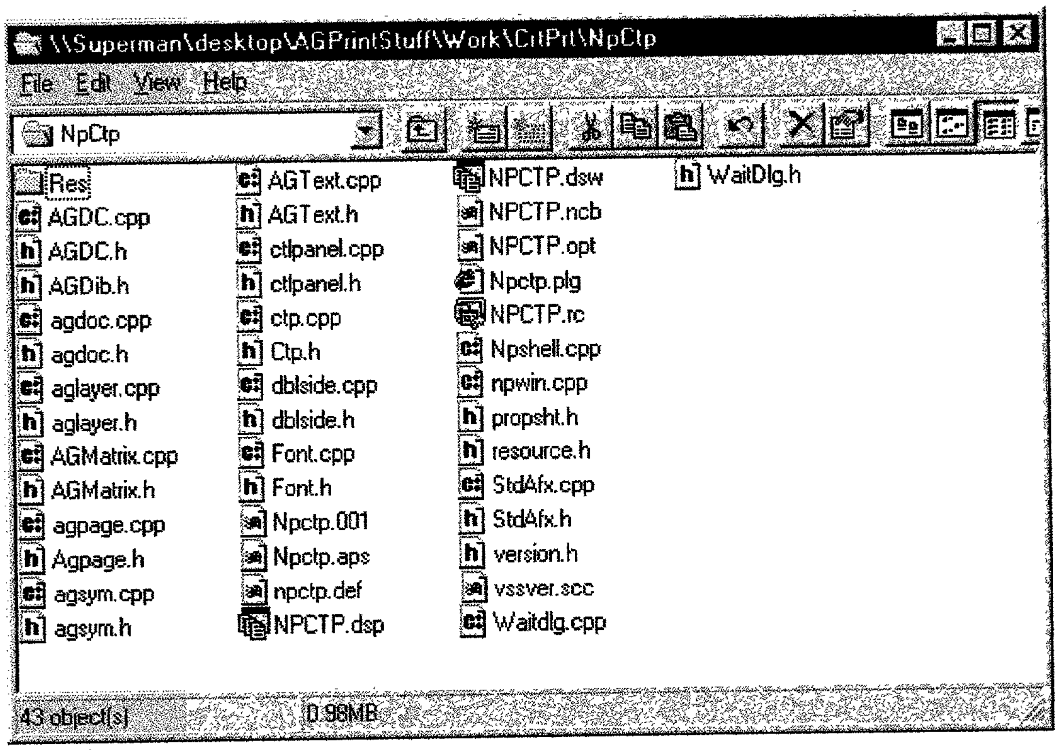


\\Superman\desktop\AGPrintStuff\Work\CrtPrt\Digital ID

agkey.pvk
ms pwd.txt
mycredentials.spc
netscape certificate.p12
netscape pwd.txt
vssver.scc

6 object(s) 13.1KB

Untitled



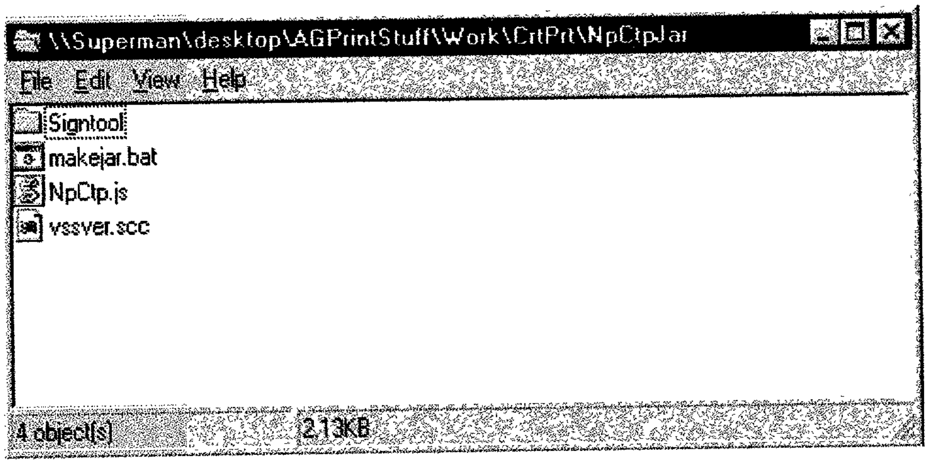
\\Superman\desktop\AGPrintStuff\Work\CtPrt\NpCtp

```
Res            AGText.cpp     NPCTP.dsw      WaitDlg.h
AGDC.cpp       AGText.h       NPCTP.ncb
AGDC.h         ctlpanel.cpp   NPCTP.opt
AGDib.h        ctlpanel.h     Npctp.plg
agdoc.cpp      ctp.cpp        NPCTP.rc
agdoc.h        Ctp.h          Npshell.cpp
aglayer.cpp    dblside.cpp    npwin.cpp
aglayer.h      dblside.h      propsht.h
AGMatrix.cpp   Font.cpp       resource.h
AGMatrix.h     Font.h         StdAfx.cpp
agpage.cpp     Npctp.001      StdAfx.h
Agpage.h       Npctp.aps      version.h
agsym.cpp      npctp.def      vssver.scc
agsym.h        NPCTP.dsp      Waitdlg.cpp
```

43 object(s)   0.98MB

[illegible]

Untitled



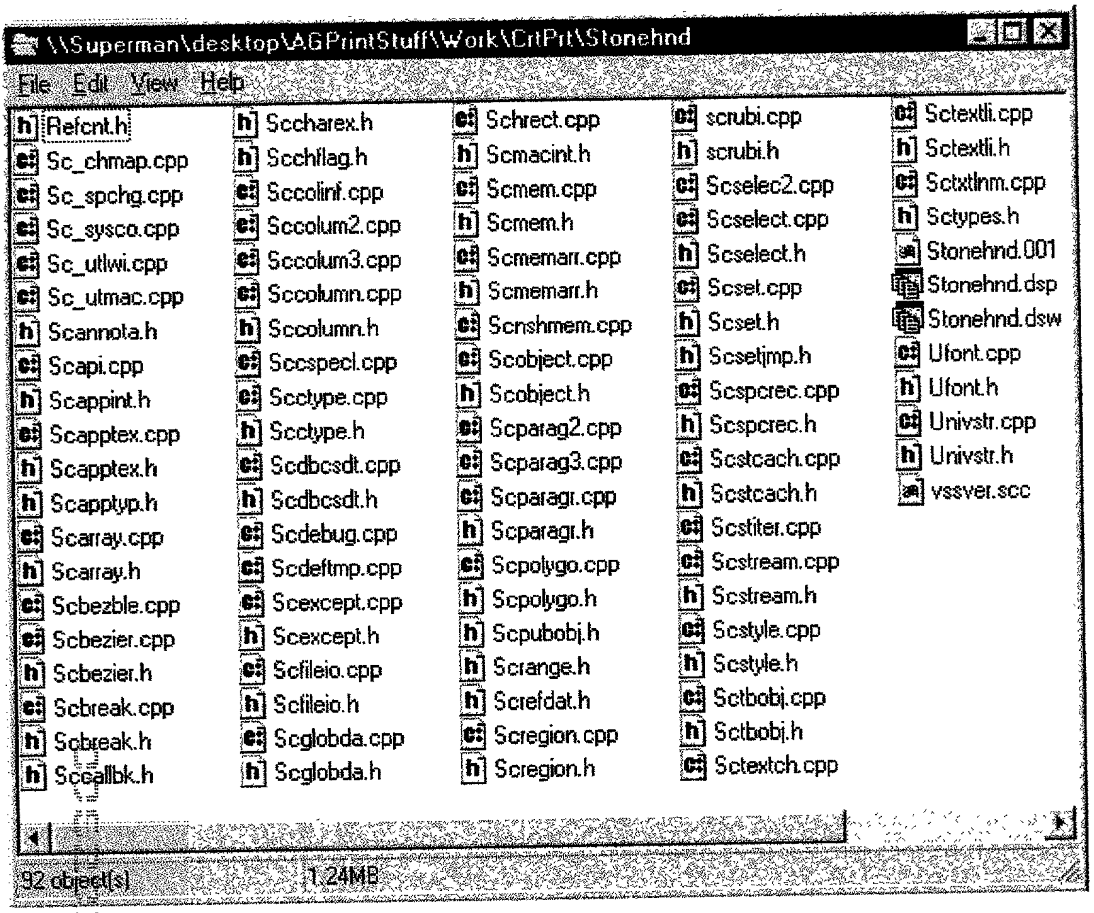
\\Superman\desktop\AGPrintStuff\Work\CrtPrt\NpCtpJar

Signtool
makejar.bat
NpCtp.js
vssver.scc

4 object(s) 2.13KB

09529370 .073400

Untitled



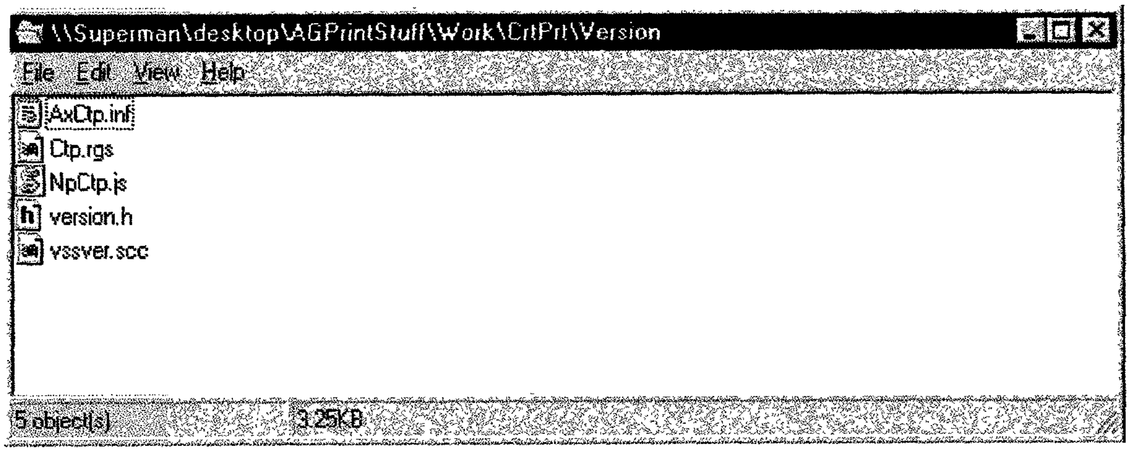
\\Superman\desktop\AGPrintStuff\Work\CrtPrt\Stonehnd

File Edit View Help

```
Refcnt.h         Sccharex.h      Schrect.cpp     scrubi.cpp      Sctextli.cpp
Sc_chmap.cpp     Scchflag.h      Scmacint.h      scrubi.h        Sctextli.h
Sc_spchg.cpp     Sccolinf.cpp    Scmem.cpp       Scselec2.cpp    Sctxtlnm.cpp
Sc_sysco.cpp     Sccolum2.cpp    Scmem.h         Scselect.cpp    Sctypes.h
Sc_utlwi.cpp     Sccolum3.cpp    Scmemarr.cpp    Scselect.h      Stonehnd.001
Sc_utmac.cpp     Sccolumn.cpp    Scmemarr.h      Scset.cpp       Stonehnd.dsp
Scannota.h       Sccolumn.h      Scnshmem.cpp    Scset.h         Stonehnd.dsw
Scapi.cpp        Sccspecl.cpp    Scobject.cpp    Scsetjmp.h      Ufont.cpp
Scappint.h       Scctype.cpp     Scobject.h      Scspcrec.cpp    Ufont.h
Scapptex.cpp     Scctype.h       Scparag2.cpp    Scspcrec.h      Univstr.cpp
Scapptex.h       Scdbcsdt.cpp    Scparag3.cpp    Scstcach.cpp    Univstr.h
Scapptyp.h       Scdbcsdt.h      Scparagr.cpp    Scstcach.h      vssver.scc
Scarray.cpp      Scdebug.cpp     Scparagr.h      Scstiter.cpp
Scarray.h        Scdeftmp.cpp    Scpolygo.cpp    Scstream.cpp
Scbezble.cpp     Scexcept.cpp    Scpolygo.h      Scstream.h
Scbezier.cpp     Scexcept.h      Scpubobj.h      Scstyle.cpp
Scbezier.h       Scfileio.cpp    Scrange.h       Scstyle.h
Scbreak.cpp      Scfileio.h      Screfdat.h      Sctbobj.cpp
Scbreak.h        Scglobda.cpp    Scregion.cpp    Sctbobj.h
Sccallbk.h       Scglobda.h      Scregion.h      Sctextch.cpp
```

92 object(s)   1.24MB

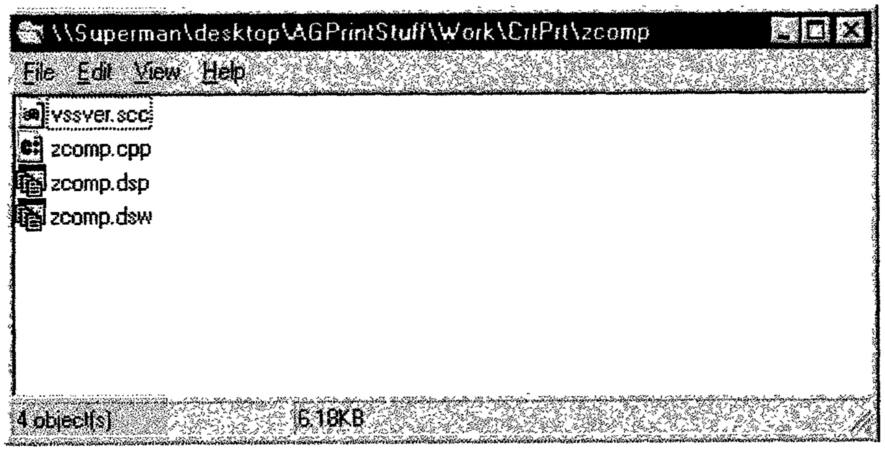
Jntitled


\\Superman\desktop\AGPrintStuff\Work\CrtPrt\Version

AxCtp.inf
Ctp.rgs
NpCtp.js
version.h
vssver.scc

5 object(s) 3.25KB

Jntitled

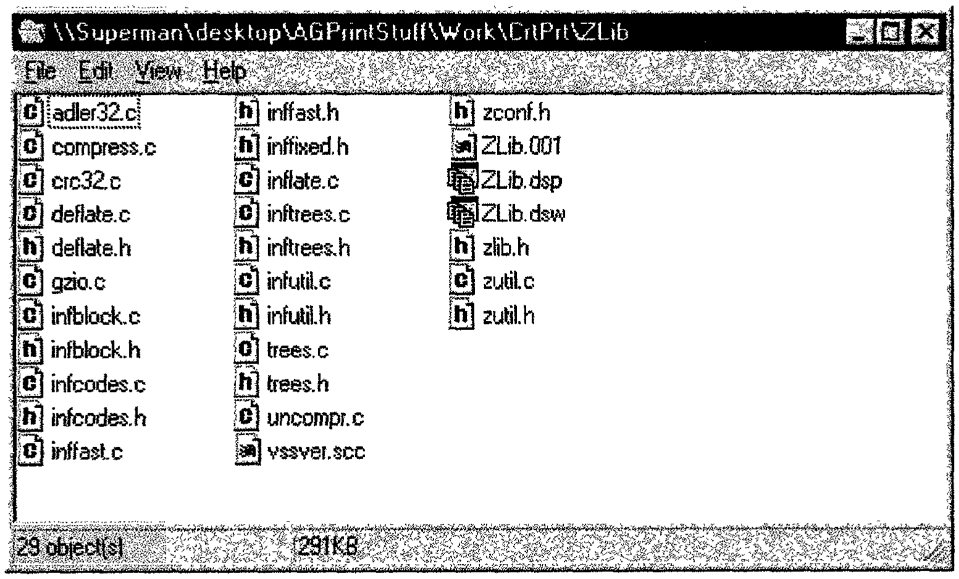


vssver.scc
zcomp.cpp
zcomp.dsp
zcomp.dsw

4 object(s) 6.18KB

09628370 .073100

Jntitled



adler32.c
compress.c
crc32.c
deflate.c
deflate.h
gzio.c
infblock.c
infblock.h
infcodes.c
infcodes.h
inffast.c
inffast.h
inffixed.h
inflate.c
inftrees.c
inftrees.h
infutil.c
infutil.h
trees.c
trees.h
uncompr.c
vssver.scc
zconf.h
ZLib.001
ZLib.dsp
ZLib.dsw
zlib.h
zutil.c
zutil.h

29 object(s) 291KB

09520370 .073100

```
/* adler32.c -- compute the Adler-32 checksum of a data stream
 * Copyright (C) 1995-1998 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* @(#) $Id$ */

#include "zlib.h"

#define BASE 65521L /* largest prime smaller than 65536 */
#define NMAX 5552
/* NMAX is the largest n such that 255n(n+1)/2 + (n+1)(BASE-1) <= 2^32-1 */

#define DO1(buf,i)  {s1 += buf[i]; s2 += s1;}
#define DO2(buf,i)  DO1(buf,i); DO1(buf,i+1);
#define DO4(buf,i)  DO2(buf,i); DO2(buf,i+2);
#define DO8(buf,i)  DO4(buf,i); DO4(buf,i+4);
#define DO16(buf)   DO8(buf,0); DO8(buf,8);

/* ========================================================================= */
uLong ZEXPORT adler32(adler, buf, len)
    uLong adler;
    const Bytef *buf;
    uInt len;
{
    unsigned long s1 = adler & 0xffff;
    unsigned long s2 = (adler >> 16) & 0xffff;
    int k;

    if (buf == Z_NULL) return 1L;

    while (len > 0) {
        k = len < NMAX ? len : NMAX;
        len -= k;
        while (k >= 16) {
            DO16(buf);
        buf += 16;
            k -= 16;
        }
        if (k != 0) do {
            s1 += *buf++;
        s2 += s1;
        } while (--k);
        s1 %= BASE;
        s2 %= BASE;
    }
    return (s2 << 16) | s1;
}
```

```
/* compress.c -- compress a memory buffer
 * Copyright (C) 1995-1998 Jean-loup Gailly.
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* @(#) $Id$ */

#include "zlib.h"

/* ===========================================================================
     Compresses the source buffer into the destination buffer. The level
   parameter has the same meaning as in deflateInit.  sourceLen is the byte
   length of the source buffer. Upon entry, destLen is the total size of the
   destination buffer, which must be at least 0.1% larger than sourceLen plus
   12 bytes. Upon exit, destLen is the actual size of the compressed buffer.

     compress2 returns Z_OK if success, Z_MEM_ERROR if there was not enough
   memory, Z_BUF_ERROR if there was not enough room in the output buffer,
   Z_STREAM_ERROR if the level parameter is invalid.
*/
int ZEXPORT compress2 (dest, destLen, source, sourceLen, level)
    Bytef *dest;
    uLongf *destLen;
    const Bytef *source;
    uLong sourceLen;
    int level;
{
    z_stream stream;
    int err;

    stream.next_in = (Bytef*)source;
    stream.avail_in = (uInt)sourceLen;
#ifdef MAXSEG_64K
    /* Check for source > 64K on 16-bit machine: */
    if ((uLong)stream.avail_in != sourceLen) return Z_BUF_ERROR;
#endif
    stream.next_out = dest;
    stream.avail_out = (uInt)*destLen;
    if ((uLong)stream.avail_out != *destLen) return Z_BUF_ERROR;

    stream.zalloc = (alloc_func)0;
    stream.zfree = (free_func)0;
    stream.opaque = (voidpf)0;

    err = deflateInit(&stream, level);
    if (err != Z_OK) return err;

    err = deflate(&stream, Z_FINISH);
    if (err != Z_STREAM_END) {
        deflateEnd(&stream);
        return err == Z_OK ? Z_BUF_ERROR : err;
    }
    *destLen = stream.total_out;

    err = deflateEnd(&stream);
    return err;
}

/* ===========================================================================
 */
int ZEXPORT compress (dest, destLen, source, sourceLen)
    Bytef *dest;
    uLongf *destLen;
    const Bytef *source;
    uLong sourceLen;
{
    return compress2(dest, destLen, source, sourceLen, Z_DEFAULT_COMPRESSION);
}
```

```
    if (buf == Z_NULL) return 0L;
#ifdef DYNAMIC_CRC_TABLE
    if (crc_table_empty)
      make_crc_table();
#endif
    crc = crc ^ 0xffffffffL;
    while (len >= 8)
    {
      DO8(buf);
      len -= 8;
    }
    if (len) do {
      DO1(buf);
    } while (--len);
    return crc ^ 0xffffffffL;
}
```

09529370.073400

```
  0x646ba8c0L, 0xfd62f97aL, 0x8a65c9ecL, 0x14015c4fL, 0x63066cd9L,
  0xfa0f3d63L, 0x8d080df5L, 0x3b6e20c8L, 0x4c69105eL, 0xd56041e4L,
  0xa2677172L, 0x3c03e4d1L, 0x4b04d447L, 0xd20d85fdL, 0xa50ab56bL,
  0x35b5a8faL, 0x42b2986cL, 0xdbbbc9d6L, 0xacbcf940L, 0x32d86ce3L,
  0x45df5c75L, 0xdcd60dcfL, 0xabd13d59L, 0x26d930acL, 0x51de003aL,
  0xc8d75180L, 0xbfd06116L, 0x21b4f4b5L, 0x56b3c423L, 0xcfba9599L,
  0xb8bda50fL, 0x2802b89eL, 0x5f058808L, 0xc60cd9b2L, 0xb10be924L,
  0x2f6f7c87L, 0x58684c11L, 0xc1611dabL, 0xb6662d3dL, 0x76dc4190L,
  0x01db7106L, 0x98d220bcL, 0xefd5102aL, 0x71b18589L, 0x06b6b51fL,
  0x9fbfe4a5L, 0xe8b8d433L, 0x7807c9a2L, 0x0f00f934L, 0x9609a88eL,
  0xe10e9818L, 0x7f6a0dbbL, 0x086d3d2dL, 0x91646c97L, 0xe6635c01L,
  0x6b6b51f4L, 0x1c6c6162L, 0x856530d8L, 0xf262004eL, 0x6c0695edL,
  0x1b01a57bL, 0x8208f4c1L, 0xf50fc457L, 0x65b0d9c6L, 0x12b7e950L,
  0x8bbeb8eaL, 0xfcb9887cL, 0x62dd1ddfL, 0x15da2d49L, 0x8cd37cf3L,
  0xfbd44c65L, 0x4db26158L, 0x3ab551ceL, 0xa3bc0074L, 0xd4bb30e2L,
  0x4adfa541L, 0x3dd895d7L, 0xa4d1c46dL, 0xd3d6f4fbL, 0x4369e96aL,
  0x346ed9fcL, 0xad678846L, 0xda60b8d0L, 0x44042d73L, 0x33031de5L,
  0xaa0a4c5fL, 0xdd0d7cc9L, 0x5005713cL, 0x270241aaL, 0xbe0b1010L,
  0xc90c2086L, 0x5768b525L, 0x206f85b3L, 0xb966d409L, 0xce61e49fL,
  0x5edef90eL, 0x29d9c998L, 0xb0d09822L, 0xc7d7a8b4L, 0x59b33d17L,
  0x2eb40d81L, 0xb7bd5c3bL, 0xc0ba6cadL, 0xedb88320L, 0x9abfb3b6L,
  0x03b6e20cL, 0x74b1d29aL, 0xead54739L, 0x9dd277afL, 0x04db2615L,
  0x73dc1683L, 0xe3630b12L, 0x94643b84L, 0x0d6d6a3eL, 0x7a6a5aa8L,
  0xe40ecf0bL, 0x9309ff9dL, 0x0a00ae27L, 0x7d079eb1L, 0xf00f9344L,
  0x8708a3d2L, 0x1e01f268L, 0x6906c2feL, 0xf762575dL, 0x806567cbL,
  0x196c3671L, 0x6e6b06e7L, 0xfed41b76L, 0x89d32be0L, 0x10da7a5aL,
  0x67dd4accL, 0xf9b9df6fL, 0x8ebeeff9L, 0x17b7be43L, 0x60b08ed5L,
  0xd6d6a3e8L, 0xa1d1937eL, 0x38d8c2c4L, 0x4fdff252L, 0xd1bb67f1L,
  0xa6bc5767L, 0x3fb506ddL, 0x48b2364bL, 0xd80d2bdaL, 0xaf0a1b4cL,
  0x36034af6L, 0x41047a60L, 0xdf60efc3L, 0xa867df55L, 0x316e8eefL,
  0x4669be79L, 0xcb61b38cL, 0xbc66831aL, 0x256fd2a0L, 0x5268e236L,
  0xcc0c7795L, 0xbb0b4703L, 0x220216b9L, 0x5505262fL, 0xc5ba3bbeL,
  0xb2bd0b28L, 0x2bb45a92L, 0x5cb36a04L, 0xc2d7ffa7L, 0xb5d0cf31L,
  0x2cd99e8bL, 0x5bdeae1dL, 0x9b64c2b0L, 0xec63f226L, 0x756aa39cL,
  0x026d930aL, 0x9c0906a9L, 0xeb0e363fL, 0x72076785L, 0x05005713L,
  0x95bf4a82L, 0xe2b87a14L, 0x7bb12baeL, 0x0cb61b38L, 0x92d28e9bL,
  0xe5d5be0dL, 0x7cdcefb7L, 0x0bdbdf21L, 0x86d3d2d4L, 0xf1d4e242L,
  0x68ddb3f8L, 0x1fda836eL, 0x81be16cdL, 0xf6b9265bL, 0x6fb077e1L,
  0x18b74777L, 0x88085ae6L, 0xff0f6a70L, 0x66063bcaL, 0x11010b5cL,
  0x8f659effL, 0xf862ae69L, 0x616bffd3L, 0x166ccf45L, 0xa00ae278L,
  0xd70dd2eeL, 0x4e048354L, 0x3903b3c2L, 0xa7672661L, 0xd06016f7L,
  0x4969474dL, 0x3e6e77dbL, 0xaed16a4aL, 0xd9d65adcL, 0x40df0b66L,
  0x37d83bf0L, 0xa9bcae53L, 0xdebb9ec5L, 0x47b2cf7fL, 0x30b5ffe9L,
  0xbdbdf21cL, 0xcabac28aL, 0x53b39330L, 0x24b4a3a6L, 0xbad03605L,
  0xcdd70693L, 0x54de5729L, 0x23d967bfL, 0xb3667a2eL, 0xc4614ab8L,
  0x5d681b02L, 0x2a6f2b94L, 0xb40bbe37L, 0xc30c8ea1L, 0x5a05df1bL,
  0x2d02ef8dL
};
#endif

/* =========================================================================
 * This function can be used by asm versions of crc32()
 */
const uLongf * ZEXPORT get_crc_table()
{
#ifdef DYNAMIC_CRC_TABLE
  if (crc_table_empty) make_crc_table();
#endif
  return (const uLongf *)crc_table;
}

/* ========================================================================= */
#define DO1(buf) crc = crc_table[((int)crc ^ (*buf++)) & 0xff] ^ (crc >> 8);
#define DO2(buf)  DO1(buf); DO1(buf);
#define DO4(buf)  DO2(buf); DO2(buf);
#define DO8(buf)  DO4(buf); DO4(buf);

/* ========================================================================= */
uLong ZEXPORT crc32(crc, buf, len)
    uLong crc;
    const Bytef *buf;
    uInt len;
{
```

```
/* crc32.c -- compute the CRC-32 of a data stream
 * Copyright (C) 1995-1998 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* @(#) $Id$ */

#include "zlib.h"

#define local static

#ifdef DYNAMIC_CRC_TABLE

local int crc_table_empty = 1;
local uLongf crc_table[256];
local void make_crc_table OF((void));

/*
  Generate a table for a byte-wise 32-bit CRC calculation on the polynomial:
  x^32+x^26+x^23+x^22+x^16+x^12+x^11+x^10+x^8+x^7+x^5+x^4+x^2+x+1.

  Polynomials over GF(2) are represented in binary, one bit per coefficient,
  with the lowest powers in the most significant bit.  Then adding polynomials
  is just exclusive-or, and multiplying a polynomial by x is a right shift by
  one.  If we call the above polynomial p, and represent a byte as the
  polynomial q, also with the lowest power in the most significant bit (so the
  byte 0xb1 is the polynomial x^7+x^3+x+1), then the CRC is (q*x^32) mod p,
  where a mod b means the remainder after dividing a by b.

  This calculation is done using the shift-register method of multiplying and
  taking the remainder.  The register is initialized to zero, and for each
  incoming bit, x^32 is added mod p to the register if the bit is a one (where
  x^32 mod p is p+x^32 = x^26+...+1), and the register is multiplied mod p by
  x (which is shifting right by one and adding x^32 mod p if the bit shifted
  out is a one).  We start with the highest power (least significant bit) of
  q and repeat for all eight bits of q.

  The table is simply the CRC of all possible eight bit values.  This is all
  the information needed to generate CRC's on data a byte at a time for all
  combinations of CRC register values and incoming bytes.
*/
local void make_crc_table()
{
  uLong c;
  int n, k;
  uLong poly;            /* polynomial exclusive-or pattern */
  /* terms of polynomial defining this crc (except x^32): */
  static const Byte p[] = {0,1,2,4,5,7,8,10,11,12,16,22,23,26};

  /* make exclusive-or pattern from polynomial (0xedb88320L) */
  poly = 0L;
  for (n = 0; n < sizeof(p)/sizeof(Byte); n++)
    poly |= 1L << (31 - p[n]);

  for (n = 0; n < 256; n++)
  {
    c = (uLong)n;
    for (k = 0; k < 8; k++)
      c = c & 1 ? poly ^ (c >> 1) : c >> 1;
    crc_table[n] = c;
  }
  crc_table_empty = 0;
}
#else
/* ========================================================================
 * Table of CRC-32's of all single-byte values (made by make_crc_table)
 */
local const uLongf crc_table[256] = {
  0x00000000L, 0x77073096L, 0xee0e612cL, 0x990951baL, 0x076dc419L,
  0x706af48fL, 0xe963a535L, 0x9e6495a3L, 0x0edb8832L, 0x79dcb8a4L,
  0xe0d5e91eL, 0x97d2d988L, 0x09b64c2bL, 0x7eb17cbdL, 0xe7b82d07L,
  0x90bf1d91L, 0x1db71064L, 0x6ab020f2L, 0xf3b97148L, 0x84be41deL,
  0x1adad47dL, 0x6ddde4ebL, 0xf4d4b551L, 0x83d385c7L, 0x136c9856L,
            s->match_length = MIN_MATCH-1;
            s->strstart++;

            if (bflush) FLUSH_BLOCK(s, 0);

        } else if (s->match_available) {
            /* If there was no match at the previous position, output a
             * single literal. If there was a match but the current match
             * is longer, truncate the previous match to a single literal.
             */
            Tracevv((stderr,"%c", s->window[s->strstart-1]));
        _tr_tally_lit(s, s->window[s->strstart-1], bflush);
        if (bflush) {
                FLUSH_BLOCK_ONLY(s, 0);
            }
            s->strstart++;
            s->lookahead--;
            if (s->strm->avail_out == 0) return need_more;
        } else {
            /* There is no previous match to compare with, wait for
             * the next step to decide.
             */
            s->match_available = 1;
            s->strstart++;
            s->lookahead--;
        }
    }
    Assert (flush != Z_NO_FLUSH, "no flush?");
    if (s->match_available) {
        Tracevv((stderr,"%c", s->window[s->strstart-1]));
        _tr_tally_lit(s, s->window[s->strstart-1], bflush);
        s->match_available = 0;
    }
    FLUSH_BLOCK(s, flush == Z_FINISH);
    return flush == Z_FINISH ? finish_done : block_done;
    /* Process the input block. */
    for (;;) {
        /* Make sure that we always have enough lookahead, except
         * at the end of the input file. We need MAX_MATCH bytes
         * for the next match, plus MIN_MATCH bytes to insert the
         * string following the next match.
         */
        if (s->lookahead < MIN_LOOKAHEAD) {
            fill_window(s);
            if (s->lookahead < MIN_LOOKAHEAD && flush == Z_NO_FLUSH) {
            return need_more;
        }
            if (s->lookahead == 0) break; /* flush the current block */
        }

        /* Insert the string window[strstart .. strstart+2] in the
         * dictionary, and set hash_head to the head of the hash chain:
         */
        if (s->lookahead >= MIN_MATCH) {
            INSERT_STRING(s, s->strstart, hash_head);
        }

        /* Find the longest match, discarding those <= prev_length.
         */
        s->prev_length = s->match_length, s->prev_match = s->match_start;
        s->match_length = MIN_MATCH-1;

        if (hash_head != NIL && s->prev_length < s->max_lazy_match &&
            s->strstart - hash_head <= MAX_DIST(s)) {
            /* To simplify the code, we prevent matches with the string
             * of window index 0 (in particular we have to avoid a match
             * of the string with itself at the start of the input file).
             */
            if (s->strategy != Z_HUFFMAN_ONLY) {
                s->match_length = longest_match (s, hash_head);
            }
            /* longest_match() sets match_start */

            if (s->match_length <= 5 && (s->strategy == Z_FILTERED ||
                 (s->match_length == MIN_MATCH &&
                  s->strstart - s->match_start > TOO_FAR))) {

                /* If prev_match is also MIN_MATCH, match_start is garbage
                 * but we will ignore the current match anyway.
                 */
                s->match_length = MIN_MATCH-1;
            }
        }
        /* If there was a match at the previous step and the current
         * match is not better, output the previous match:
         */
        if (s->prev_length >= MIN_MATCH && s->match_length <= s->prev_length) {
            uInt max_insert = s->strstart + s->lookahead - MIN_MATCH;
            /* Do not insert strings in hash table beyond this. */

            check_match(s, s->strstart-1, s->prev_match, s->prev_length);

            _tr_tally_dist(s, s->strstart -1 - s->prev_match,
               s->prev_length - MIN_MATCH, bflush);

            /* Insert in hash table all strings up to the end of the match.
             * strstart-1 and strstart are already inserted. If there is not
             * enough lookahead, the last two strings are not inserted in
             * the hash table.
             */
            s->lookahead -= s->prev_length-1;
            s->prev_length -= 2;
            do {
                if (++s->strstart <= max_insert) {
                    INSERT_STRING(s, s->strstart, hash_head);
                }
            } while (--s->prev_length != 0);
            s->match_available = 0;
         */
        if (hash_head != NIL && s->strstart - hash_head <= MAX_DIST(s)) {
            /* To simplify the code, we prevent matches with the string
             * of window index 0 (in particular we have to avoid a match
             * of the string with itself at the start of the input file).
             */
            if (s->strategy != Z_HUFFMAN_ONLY) {
                s->match_length = longest_match (s, hash_head);
            }
            /* longest_match() sets match_start */
        }
        if (s->match_length >= MIN_MATCH) {
            check_match(s, s->strstart, s->match_start, s->match_length);

            _tr_tally_dist(s, s->strstart - s->match_start,
                           s->match_length - MIN_MATCH, bflush);

            s->lookahead -= s->match_length;

            /* Insert new strings in the hash table only if the match length
             * is not too large. This saves time but degrades compression.
             */
#ifndef FASTEST
            if (s->match_length <= s->max_insert_length &&
                s->lookahead >= MIN_MATCH) {
                s->match_length--; /* string at strstart already in hash table */
                do {
                    s->strstart++;
                    INSERT_STRING(s, s->strstart, hash_head);
                    /* strstart never exceeds WSIZE-MAX_MATCH, so there are
                     * always MIN_MATCH bytes ahead.
                     */
                } while (--s->match_length != 0);
                s->strstart++;
            } else
#endif
        {
                s->strstart += s->match_length;
                s->match_length = 0;
                s->ins_h = s->window[s->strstart];
                UPDATE_HASH(s, s->ins_h, s->window[s->strstart+1]);
#if MIN_MATCH != 3
                Call UPDATE_HASH() MIN_MATCH-3 more times
#endif
                /* If lookahead < MIN_MATCH, ins_h is garbage, but it does not
                 * matter since it will be recomputed at next deflate call.
                 */
            }
        } else {
            /* No match, output a literal byte */
            Tracevv((stderr,"%c", s->window[s->strstart]));
            _tr_tally_lit (s, s->window[s->strstart], bflush);
            s->lookahead--;
            s->strstart++;
        }
        if (bflush) FLUSH_BLOCK(s, 0);
    }
    FLUSH_BLOCK(s, flush == Z_FINISH);
    return flush == Z_FINISH ? finish_done : block_done;
}

/* ===========================================================================
 * Same as above, but achieves better compression. We use a lazy
 * evaluation for matches: a match is finally adopted only if there is
 * no better match at the next window position.
 */
local block_state deflate_slow(s, flush)
    deflate_state *s;
    int flush;
{
    IPos hash_head = NIL;    /* head of hash chain */
    int bflush;              /* set if current block must be flushed */
    for (;;) {
        /* Fill the window as much as possible: */
        if (s->lookahead <= 1) {

            Assert(s->strstart < s->w_size+MAX_DIST(s) ||
           s->block_start >= (long)s->w_size, "slide too late");

            fill_window(s);
            if (s->lookahead == 0 && flush == Z_NO_FLUSH) return need_more;

            if (s->lookahead == 0) break; /* flush the current block */
        }
    Assert(s->block_start >= 0L, "block gone");

    s->strstart += s->lookahead;
    s->lookahead = 0;

    /* Emit a stored block if pending_buf will be full: */
    max_start = s->block_start + max_block_size;
        if (s->strstart == 0 || (ulg)s->strstart >= max_start) {
        /* strstart == 0 is possible when wraparound on 16-bit machine */
        s->lookahead = (uInt)(s->strstart - max_start);
        s->strstart = (uInt)max_start;
            FLUSH_BLOCK(s, 0);
    }
    /* Flush if we may have to slide, otherwise block_start may become
         * negative and the data will be gone:
         */
        if (s->strstart - (uInt)s->block_start >= MAX_DIST(s)) {
            FLUSH_BLOCK(s, 0);
    }
    }
    FLUSH_BLOCK(s, flush == Z_FINISH);
    return flush == Z_FINISH ? finish_done : block_done;

/* ===========================================================================
 * Compress as much as possible from the input stream, return the current
 * block state.
 * This function does not perform lazy evaluation of matches and inserts
 * new strings in the dictionary only for unmatched strings or for short
 * matches. It is used only for the fast compression options.
 */
local block_state deflate_fast(s, flush)
    deflate_state *s;
    int flush;

    IPos hash_head = NIL; /* head of the hash chain */
    int bflush;           /* set if current block must be flushed */

    for (;;) {
        /* Make sure that we always have enough lookahead, except
         * at the end of the input file. We need MAX_MATCH bytes
         * for the next match, plus MIN_MATCH bytes to insert the
         * string following the next match.
         */
        if (s->lookahead < MIN_LOOKAHEAD) {
            fill_window(s);
            if (s->lookahead < MIN_LOOKAHEAD && flush == Z_NO_FLUSH) {
            return need_more;
        }
            if (s->lookahead == 0) break; /* flush the current block */
        }

        /* Insert the string window[strstart .. strstart+2] in the
         * dictionary, and set hash_head to the head of the hash chain:
         */
        if (s->lookahead >= MIN_MATCH) {
            INSERT_STRING(s, s->strstart, hash_head);
        }

        /* Find the longest match, discarding those <= prev_length.
         * At this point we have always match_length < MIN_MATCH
         * => more >= window_size - (MIN_LOOKAHEAD-1 + WSIZE + MAX_DIST-1)
         * => more >= window_size - 2*WSIZE + 2
         * In the BIG_MEM or MMAP case (not yet supported),
         *   window_size == input_size + MIN_LOOKAHEAD  &&
         *   strstart + s->lookahead <= input_size => more >= MIN_LOOKAHEAD.
         * Otherwise, window_size == 2*WSIZE so more >= 2.
         * If there was sliding, more >= WSIZE. So in all cases, more >= 2.
         */
        Assert(more >= 2, "more < 2");

        n = read_buf(s->strm, s->window + s->strstart + s->lookahead, more);
        s->lookahead += n;

        /* Initialize the hash value now that we have some input: */
        if (s->lookahead >= MIN_MATCH) {
            s->ins_h = s->window[s->strstart];
            UPDATE_HASH(s, s->ins_h, s->window[s->strstart+1]);
#if MIN_MATCH != 3
            Call UPDATE_HASH() MIN_MATCH-3 more times
#endif
        }
        /* If the whole input has less than MIN_MATCH bytes, ins_h is garbage,
         * but this is not important since only literal bytes will be emitted.
         */

    } while (s->lookahead < MIN_LOOKAHEAD && s->strm->avail_in != 0);
}

/* ===========================================================================
 * Flush the current block, with given end-of-file flag.
 * IN assertion: strstart is set to the end of the current match.
 */
#define FLUSH_BLOCK_ONLY(s, eof) { \
   _tr_flush_block(s, (s->block_start >= 0L ? \
                   (charf *)&s->window[(unsigned)s->block_start] : \
                   (charf *)Z_NULL), \
                (ulg)((long)s->strstart - s->block_start), \
                (eof)); \
   s->block_start = s->strstart; \
   flush_pending(s->strm); \
   Tracev((stderr,"[FLUSH]")); \
}

/* Same but force premature exit if necessary. */
#define FLUSH_BLOCK(s, eof) { \
   FLUSH_BLOCK_ONLY(s, eof); \
   if (s->strm->avail_out == 0) return (eof) ? finish_started : need_more; \
}

/* ===========================================================================
 * Copy without compression as much as possible from the input stream, return
 * the current block state.
 * This function does not insert new strings in the dictionary since
 * uncompressible data is probably not useful. This function is used
 * only for the level=0 compression option.
 * NOTE: this function should be optimized to avoid extra copying from
 * window to pending_buf.
 */
local block_state deflate_stored(s, flush)
    deflate_state *s;
    int flush;
{
    /* Stored blocks are limited to 0xffff bytes, pending_buf is limited
     * to pending_buf_size, and each stored block has a 5 byte header:
     */
    ulg max_block_size = 0xffff;
    ulg max_start;

    if (max_block_size > s->pending_buf_size - 5) {
        max_block_size = s->pending_buf_size - 5;
    }

    /* Copy as much as possible from input to output: */
/* ===========================================================================
 * Fill the window when the lookahead becomes insufficient.
 * Updates strstart and lookahead.
 *
 * IN assertion: lookahead < MIN_LOOKAHEAD
 * OUT assertions: strstart <= window_size-MIN_LOOKAHEAD
 *    At least one byte has been read, or avail_in == 0; reads are
 *    performed for at least two bytes (required for the zip translate_eol
 *    option -- not supported here).
 */
local void fill_window(s)
    deflate_state *s;
{
    register unsigned n, m;
    register Posf *p;
    unsigned more;    /* Amount of free space at the end of the window. */
    uInt wsize = s->w_size;

    do {
        more = (unsigned)(s->window_size -(ulg)s->lookahead -(ulg)s->strstart);

        /* Deal with !@#$% 64K limit: */
        if (more == 0 && s->strstart == 0 && s->lookahead == 0) {
            more = wsize;

        } else if (more == (unsigned)(-1)) {
            /* Very unlikely, but possible on 16 bit machine if strstart == 0
             * and lookahead == 1 (input done one byte at time)
             */
            more--;

        /* If the window is almost full and there is insufficient lookahead,
         * move the upper half to the lower one to make room in the upper half.
         */
        } else if (s->strstart >= wsize+MAX_DIST(s)) {

            zmemcpy(s->window, s->window+wsize, (unsigned)wsize);
            s->match_start -= wsize;
            s->strstart    -= wsize; /* we now have strstart >= MAX_DIST */
            s->block_start -= (long) wsize;

            /* Slide the hash table (could be avoided with 32 bit values
               at the expense of memory usage). We slide even when level == 0
               to keep the hash table consistent if we switch back to level > 0
               later. (Using level 0 permanently is not an optimal usage of
               zlib, so we don't care about this pathological case.)
             */
        n = s->hash_size;
        p = &s->head[n];
        do {
        m = *--p;
        *p = (Pos)(m >= wsize ? m-wsize : NIL);
        } while (--n);

        n = wsize;
#ifndef FASTEST
        p = &s->prev[n];
        do {
        m = *--p;
        *p = (Pos)(m >= wsize ? m-wsize : NIL);
        /* If n is not on any hash chain, prev[n] is garbage but
         * its value will never be used.
         */
        } while (--n);
#endif
            more += wsize;
        }
        if (s->strm->avail_in == 0) return;

        /* If there was no sliding:
         *    strstart <= WSIZE+MAX_DIST-1 && lookahead <= MIN_LOOKAHEAD - 1 &&
         *    more == window_size - lookahead - strstart
    /* The code is optimized for HASH_BITS >= 8 and MAX_MATCH-2 multiple of 16.
     * It is easy to get rid of this optimization if necessary.
     */
    Assert(s->hash_bits >= 8 && MAX_MATCH == 258, "Code too clever");

    Assert((ulg)s->strstart <= s->window_size-MIN_LOOKAHEAD, "need lookahead");

    Assert(cur_match < s->strstart, "no future");

    match = s->window + cur_match;

    /* Return failure if the match length is less than 2:
     */
    if (match[0] != scan[0] || match[1] != scan[1]) return MIN_MATCH-1;

    /* The check at best_len-1 can be removed because it will be made
     * again later. (This heuristic is not always a win.)
     * It is not necessary to compare scan[2] and match[2] since they
     * are always equal when the other bytes match, given that
     * the hash keys are equal and that HASH_BITS >= 8.
     */
    scan += 2, match += 2;
    Assert(*scan == *match, "match[2]?");

    /* We check for insufficient lookahead only every 8th comparison;
     * the 256th check will be made at strstart+258.
     */
    do {
    } while (*++scan == *++match && *++scan == *++match &&
             *++scan == *++match && *++scan == *++match &&
             *++scan == *++match && *++scan == *++match &&
             *++scan == *++match && *++scan == *++match &&
             scan < strend);

    Assert(scan <= s->window+(unsigned)(s->window_size-1), "wild scan");

    len = MAX_MATCH - (int)(strend - scan);

    if (len < MIN_MATCH) return MIN_MATCH - 1;

    s->match_start = cur_match;
    return len <= s->lookahead ? len : s->lookahead;
}
#endif /* FASTEST */
#endif /* ASMV */

#ifdef DEBUG
/* ===========================================================================
 * Check that the match at match_start is indeed a match.
 */
local void check_match(s, start, match, length)
    deflate_state *s;
    IPos start, match;
    int length;
{
    /* check that the match is indeed a match */
    if (zmemcmp(s->window + match,
                s->window + start, length) != EQUAL) {
        fprintf(stderr, " start %u, match %u, length %d\n",
        start, match, length);
        do {
        fprintf(stderr, "%c%c", s->window[match++], s->window[start++]);
    } while (--length != 0);
        z_error("invalid match");
    }
    if (z_verbose > 1) {
        fprintf(stderr,"\\[%d,%d]", start-match, length);
        do { putc(s->window[start++], stderr); } while (--length != 0);
    }
}
#else
#  define check_match(s, start, match, length)
#endif
```

```
        /* The funny "do {}" generates better code on most compilers */

        /* Here, scan <= window+strstart+257 */
        Assert(scan <= s->window+(unsigned)(s->window_size-1), "wild scan");
        if (*scan == *match) scan++;

        len = (MAX_MATCH - 1) - (int)(strend-scan);
        scan = strend - (MAX_MATCH-1);

#else /* UNALIGNED_OK */

        if (match[best_len]   != scan_end  ||
            match[best_len-1] != scan_end1 ||
            *match            != *scan     ||
            *++match          != scan[1])      continue;

        /* The check at best_len-1 can be removed because it will be made
         * again later. (This heuristic is not always a win.)
         * It is not necessary to compare scan[2] and match[2] since they
         * are always equal when the other bytes match, given that
         * the hash keys are equal and that HASH_BITS >= 8.
         */
        scan += 2, match++;
        Assert(*scan == *match, "match[2]?");

        /* We check for insufficient lookahead only every 8th comparison;
         * the 256th check will be made at strstart+258.
         */
        do {
        } while (*++scan == *++match && *++scan == *++match &&
                 *++scan == *++match && *++scan == *++match &&
                 *++scan == *++match && *++scan == *++match &&
                 *++scan == *++match && *++scan == *++match &&
                 scan < strend);

        Assert(scan <= s->window+(unsigned)(s->window_size-1), "wild scan");

        len = MAX_MATCH - (int)(strend - scan);
        scan = strend - MAX_MATCH;

#endif /* UNALIGNED_OK */

        if (len > best_len) {
            s->match_start = cur_match;
            best_len = len;
            if (len >= nice_match) break;
#ifdef UNALIGNED_OK
            scan_end = *(ushf*)(scan+best_len-1);
#else
            scan_end1  = scan[best_len-1];
            scan_end   = scan[best_len];
#endif
        }
    } while ((cur_match = prev[cur_match & wmask]) > limit
             && --chain_length != 0);

    if ((uInt)best_len <= s->lookahead) return (uInt)best_len;
    return s->lookahead;
}

#else /* FASTEST */
/* ---------------------------------------------------------------------------
 * Optimized version for level == 1 only
 */
local uInt longest_match(s, cur_match)
    deflate_state *s;
    IPos cur_match;                             /* current match */
{
    register Bytef *scan = s->window + s->strstart; /* current string */
    register Bytef *match;                       /* matched string */
    register int len;                           /* length of current match */
    register Bytef *strend = s->window + s->strstart + MAX_MATCH;
    register Bytef *scan = s->window + s->strstart; /* current string */
    register Bytef *match;                       /* matched string */
    register int len;                           /* length of current match */
    int best_len = s->prev_length;              /* best match length so far */
    int nice_match = s->nice_match;             /* stop if match long enough */
    IPos limit = s->strstart > (IPos)MAX_DIST(s) ?
        s->strstart - (IPos)MAX_DIST(s) : NIL;
    /* Stop when cur_match becomes <= limit. To simplify the code,
     * we prevent matches with the string of window index 0.
     */
    Posf *prev = s->prev;
    uInt wmask = s->w_mask;

#ifdef UNALIGNED_OK
    /* Compare two bytes at a time. Note: this is not always beneficial.
     * Try with and without -DUNALIGNED_OK to check.
     */
    register Bytef *strend = s->window + s->strstart + MAX_MATCH - 1;
    register ush scan_start = *(ushf*)scan;
    register ush scan_end   = *(ushf*)(scan+best_len-1);
#else
    register Bytef *strend = s->window + s->strstart + MAX_MATCH;
    register Byte scan_end1  = scan[best_len-1];
    register Byte scan_end   = scan[best_len];
#endif

    /* The code is optimized for HASH_BITS >= 8 and MAX_MATCH-2 multiple of 16.
     * It is easy to get rid of this optimization if necessary.
     */
    Assert(s->hash_bits >= 8 && MAX_MATCH == 258, "Code too clever");

    /* Do not waste too much time if we already have a good match: */
    if (s->prev_length >= s->good_match) {
        chain_length >>= 2;
    }
    /* Do not look for matches beyond the end of the input. This is necessary
     * to make deflate deterministic.
     */
    if ((uInt)nice_match > s->lookahead) nice_match = s->lookahead;

    Assert((ulg)s->strstart <= s->window_size-MIN_LOOKAHEAD, "need lookahead");

    do {
        Assert(cur_match < s->strstart, "no future");
        match = s->window + cur_match;

        /* Skip to next match if the match length cannot increase
         * or if the match length is less than 2:
         */
#if (defined(UNALIGNED_OK) && MAX_MATCH == 258)
        /* This code assumes sizeof(unsigned short) == 2. Do not use
         * UNALIGNED_OK if your compiler uses a different size.
         */
        if (*(ushf*)(match+best_len-1) != scan_end ||
            *(ushf*)match != scan_start) continue;

        /* It is not necessary to compare scan[2] and match[2] since they are
         * always equal when the other bytes match, given that the hash keys
         * are equal and that HASH_BITS >= 8. Compare 2 bytes at a time at
         * strstart+3, +5, ... up to strstart+257. We check for insufficient
         * lookahead only every 4th comparison; the 128th check will be made
         * at strstart+257. If MAX_MATCH-2 is not a multiple of 8, it is
         * necessary to put more guard bytes at the end of the window, or
         * to check more often for insufficient lookahead.
         */
        Assert(scan[2] == match[2], "scan[2]?");
        scan++, match++;
        do {
        } while (*(ushf*)(scan+=2) == *(ushf*)(match+=2) &&
                 *(ushf*)(scan+=2) == *(ushf*)(match+=2) &&
                 *(ushf*)(scan+=2) == *(ushf*)(match+=2) &&
                 *(ushf*)(scan+=2) == *(ushf*)(match+=2) &&
                 scan < strend);
```

```
 * this function so some applications may wish to modify it to avoid
 * allocating a large strm->next_in buffer and copying from it.
 * (See also flush_pending()).
 */
local int read_buf(strm, buf, size)
    z_streamp strm;
    Bytef *buf;
    unsigned size;
{
    unsigned len = strm->avail_in;

    if (len > size) len = size;
    if (len == 0) return 0;

    strm->avail_in  -= len;

    if (!strm->state->noheader) {
        strm->adler = adler32(strm->adler, strm->next_in, len);
    }
    zmemcpy(buf, strm->next_in, len);
    strm->next_in  += len;
    strm->total_in += len;

    return (int)len;
}

/* ===========================================================================
 * Initialize the "longest match" routines for a new zlib stream
 */
local void lm_init (s)
    deflate_state *s;
{
    s->window_size = (ulg)2L*s->w_size;

    CLEAR_HASH(s);

    /* Set the default configuration parameters:
     */
    s->max_lazy_match   = configuration_table[s->level].max_lazy;
    s->good_match       = configuration_table[s->level].good_length;
    s->nice_match       = configuration_table[s->level].nice_length;
    s->max_chain_length = configuration_table[s->level].max_chain;

    s->strstart = 0;
    s->block_start = 0L;
    s->lookahead = 0;
    s->match_length = s->prev_length = MIN_MATCH-1;
    s->match_available = 0;
    s->ins_h = 0;
#ifdef ASMV
    match_init(); /* initialize the asm code */
#endif
}

/* ===========================================================================
 * Set match_start to the longest match starting at the given string and
 * return its length. Matches shorter or equal to prev_length are discarded,
 * in which case the result is equal to prev_length and match_start is
 * garbage.
 * IN assertions: cur_match is the head of the hash chain for the current
 *   string (strstart) and its distance is <= MAX_DIST, and prev_length >= 1
 * OUT assertion: the match length is not greater than s->lookahead.
 */
#ifndef ASMV
/* For 80x86 and 680x0, an optimized version will be provided in match.asm or
 * match.S. The code will be functionally equivalent.
 */
#ifndef FASTEST
local uInt longest_match(s, cur_match)
    deflate_state *s;
    IPos cur_match;                             /* current match */
{
    unsigned chain_length = s->max_chain_length;/* max hash chain length */
    TRY_FREE(strm, strm->state->head);
    TRY_FREE(strm, strm->state->prev);
    TRY_FREE(strm, strm->state->window);

    ZFREE(strm, strm->state);
    strm->state = Z_NULL;

    return status == BUSY_STATE ? Z_DATA_ERROR : Z_OK;
}

/* =========================================================================
 * Copy the source state to the destination state.
 * To simplify the source, this is not supported for 16-bit MSDOS (which
 * doesn't have enough memory anyway to duplicate compression states).
 */
int ZEXPORT deflateCopy (dest, source)
    z_streamp dest;
    z_streamp source;
{
#ifdef MAXSEG_64K
    return Z_STREAM_ERROR;
#else
    deflate_state *ds;
    deflate_state *ss;
    ushf *overlay;


    if (source == Z_NULL || dest == Z_NULL || source->state == Z_NULL) {
        return Z_STREAM_ERROR;
    }

    ss = source->state;

    *dest = *source;

    ds = (deflate_state *) ZALLOC(dest, 1, sizeof(deflate_state));
    if (ds == Z_NULL) return Z_MEM_ERROR;
    dest->state = (struct internal_state FAR *) ds;
    *ds = *ss;
    ds->strm = dest;

    ds->window = (Bytef *) ZALLOC(dest, ds->w_size, 2*sizeof(Byte));
    ds->prev   = (Posf *)  ZALLOC(dest, ds->w_size, sizeof(Pos));
    ds->head   = (Posf *)  ZALLOC(dest, ds->hash_size, sizeof(Pos));
    overlay = (ushf *) ZALLOC(dest, ds->lit_bufsize, sizeof(ush)+2);
    ds->pending_buf = (uchf *) overlay;

    if (ds->window == Z_NULL || ds->prev == Z_NULL || ds->head == Z_NULL ||
        ds->pending_buf == Z_NULL) {
        deflateEnd (dest);
        return Z_MEM_ERROR;
    }
    /* following zmemcpy do not work for 16-bit MSDOS */
    zmemcpy(ds->window, ss->window, ds->w_size * 2 * sizeof(Byte));
    zmemcpy(ds->prev, ss->prev, ds->w_size * sizeof(Pos));
    zmemcpy(ds->head, ss->head, ds->hash_size * sizeof(Pos));
    zmemcpy(ds->pending_buf, ss->pending_buf, (uInt)ds->pending_buf_size);

    ds->pending_out = ds->pending_buf + (ss->pending_out - ss->pending_buf);
    ds->d_buf = overlay + ds->lit_bufsize/sizeof(ush);
    ds->l_buf = ds->pending_buf + (1+sizeof(ush))*ds->lit_bufsize;

    ds->l_desc.dyn_tree = ds->dyn_ltree;
    ds->d_desc.dyn_tree = ds->dyn_dtree;
    ds->bl_desc.dyn_tree = ds->bl_tree;

    return Z_OK;
#endif
}

/* =========================================================================
 * Read a new buffer from the current input stream, update the adler32
 * and total number of bytes read.  All deflate() input goes through
```

```
    if (strm->avail_in != 0 || s->lookahead != 0 ||
        (flush != Z_NO_FLUSH && s->status != FINISH_STATE)) {
        block_state bstate;

    bstate = (*(configuration_table[s->level].func))(s, flush);

        if (bstate == finish_started || bstate == finish_done) {
            s->status = FINISH_STATE;
        }
        if (bstate == need_more || bstate == finish_started) {
        if (strm->avail_out == 0) {
            s->last_flush = -1; /* avoid BUF_ERROR next call, see above */
        }
        return Z_OK;
        /* If flush != Z_NO_FLUSH && avail_out == 0, the next call
         * of deflate should use the same flush parameter to make sure
         * that the flush is complete. So we don't have to output an
         * empty block here, this will be done at next call. This also
         * ensures that for a very small output buffer, we emit at most
         * one empty block.
         */
     }
        if (bstate == block_done) {
            if (flush == Z_PARTIAL_FLUSH) {
                _tr_align(s);
            } else { /* FULL_FLUSH or SYNC_FLUSH */
                _tr_stored_block(s, (char*)0, 0L, 0);
                /* For a full flush, this empty block will be recognized
                 * as a special marker by inflate_sync().
                 */
                if (flush == Z_FULL_FLUSH) {
                    CLEAR_HASH(s);             /* forget history */
                }
            }
            flush_pending(strm);
        if (strm->avail_out == 0) {
          s->last_flush = -1; /* avoid BUF_ERROR at next call, see above */
          return Z_OK;
        }
        }
    }
    Assert(strm->avail_out > 0, "bug2");

    if (flush != Z_FINISH) return Z_OK;
    if (s->noheader) return Z_STREAM_END;

    /* Write the zlib trailer (adler32) */
    putShortMSB(s, (uInt)(strm->adler >> 16));
    putShortMSB(s, (uInt)(strm->adler & 0xffff));
    flush_pending(strm);
    /* If avail_out is zero, the application will call deflate again
     * to flush the rest.
     */
    s->noheader = -1; /* write the trailer only once! */
    return s->pending != 0 ? Z_OK : Z_STREAM_END;
}

/* ========================================================================= */
int ZEXPORT deflateEnd (strm)
    z_streamp strm;
{
    int status;

    if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;

    status = strm->state->status;
    if (status != INIT_STATE && status != BUSY_STATE &&
    status != FINISH_STATE) {
      return Z_STREAM_ERROR;
    }

    /* Deallocate in reverse order of allocations: */
    TRY_FREE(strm, strm->state->pending_buf);
```

```
{
    int old_flush; /* value of flush param for previous deflate call */
    deflate_state *s;

    if (strm == Z_NULL || strm->state == Z_NULL ||
    flush > Z_FINISH || flush < 0) {
        return Z_STREAM_ERROR;
    }
    s = strm->state;

    if (strm->next_out == Z_NULL ||
        (strm->next_in == Z_NULL && strm->avail_in != 0) ||
    (s->status == FINISH_STATE && flush != Z_FINISH)) {
        ERR_RETURN(strm, Z_STREAM_ERROR);
    }
    if (strm->avail_out == 0) ERR_RETURN(strm, Z_BUF_ERROR);

    s->strm = strm; /* just in case */
    old_flush = s->last_flush;
    s->last_flush = flush;

    /* Write the zlib header */
    if (s->status == INIT_STATE) {

        uInt header = (Z_DEFLATED + ((s->w_bits-8)<<4)) << 8;
        uInt level_flags = (s->level-1) >> 1;

        if (level_flags > 3) level_flags = 3;
        header |= (level_flags << 6);
    if (s->strstart != 0) header |= PRESET_DICT;
        header += 31 - (header % 31);

        s->status = BUSY_STATE;
        putShortMSB(s, header);

    /* Save the adler32 of the preset dictionary: */
    if (s->strstart != 0) {
        putShortMSB(s, (uInt)(strm->adler >> 16));
        putShortMSB(s, (uInt)(strm->adler & 0xffff));
    }
    strm->adler = 1L;
    }

    /* Flush as much pending output as possible */
    if (s->pending != 0) {
        flush_pending(strm);
        if (strm->avail_out == 0) {
        /* Since avail_out is 0, deflate will be called again with
         * more output space, but possibly with both pending and
         * avail_in equal to zero. There won't be anything to do,
         * but this is not an error situation so make sure we
         * return OK instead of BUF_ERROR at next call of deflate:
             */
        s->last_flush = -1;
        return Z_OK;
    }

    /* Make sure there is something to do and avoid duplicate consecutive
     * flushes. For repeated and useless calls with Z_FINISH, we keep
     * returning Z_STREAM_END instead of Z_BUFF_ERROR.
     */
    } else if (strm->avail_in == 0 && flush <= old_flush &&
           flush != Z_FINISH) {
        ERR_RETURN(strm, Z_BUF_ERROR);
    }

    /* User must not provide more input after the first FINISH: */
    if (s->status == FINISH_STATE && strm->avail_in != 0) {
        ERR_RETURN(strm, Z_BUF_ERROR);
    }

    /* Start a new block or continue the current one.
     */
```

```
{
    deflate_state *s;
    compress_func func;
    int err = Z_OK;

    if (strm == Z_NULL || strm->state == Z_NULL) return Z_STREAM_ERROR;
    s = strm->state;

    if (level == Z_DEFAULT_COMPRESSION) {
    level = 6;
    }
    if (level < 0 || level > 9 || strategy < 0 || strategy > Z_HUFFMAN_ONLY) {
    return Z_STREAM_ERROR;
    }
    func = configuration_table[s->level].func;

    if (func != configuration_table[level].func && strm->total_in != 0) {
    /* Flush the last buffer: */
    err = deflate(strm, Z_PARTIAL_FLUSH);
    }
    if (s->level != level) {
    s->level = level;
    s->max_lazy_match   = configuration_table[level].max_lazy;
    s->good_match       = configuration_table[level].good_length;
    s->nice_match       = configuration_table[level].nice_length;
    s->max_chain_length = configuration_table[level].max_chain;
    }
    s->strategy = strategy;
    return err;


/* ===========================================================================
 * Put a short in the pending buffer. The 16-bit value is put in MSB order.
 * IN assertion: the stream state is correct and there is enough room in
 * pending_buf.
 */
local void putShortMSB (s, b)
    deflate_state *s;
    uInt b;
{
    put_byte(s, (Byte)(b >> 8));
    put_byte(s, (Byte)(b & 0xff));


/* ===========================================================================
 * Flush as much pending output as possible. All deflate() output goes
 * through this function so some applications may wish to modify it
 * to avoid allocating a large strm->next_out buffer and copying into it.
 * (See also read_buf()).
 */
local void flush_pending(strm)
    z_streamp strm;
{
    unsigned len = strm->state->pending;

    if (len > strm->avail_out) len = strm->avail_out;
    if (len == 0) return;

    zmemcpy(strm->next_out, strm->state->pending_out, len);
    strm->next_out  += len;
    strm->state->pending_out  += len;
    strm->total_out += len;
    strm->avail_out  -= len;
    strm->state->pending -= len;
    if (strm->state->pending == 0) {
        strm->state->pending_out = strm->state->pending_buf;
    }
}

/* =========================================================================== */
int ZEXPORT deflate (strm, flush)
    z_streamp strm;
    int flush;
    const Bytef *dictionary;
    uInt  dictLength;
{
    deflate_state *s;
    uInt length = dictLength;
    uInt n;
    IPos hash_head = 0;

    if (strm == Z_NULL || strm->state == Z_NULL || dictionary == Z_NULL ||
        strm->state->status != INIT_STATE) return Z_STREAM_ERROR;

    s = strm->state;
    strm->adler = adler32(strm->adler, dictionary, dictLength);

    if (length < MIN_MATCH) return Z_OK;
    if (length > MAX_DIST(s)) {
    length = MAX_DIST(s);
#ifndef USE_DICT_HEAD
    dictionary += dictLength - length; /* use the tail of the dictionary */
#endif
    }
    zmemcpy(s->window, dictionary, length);
    s->strstart = length;
    s->block_start = (long)length;

    /* Insert all strings in the hash table (except for the last two bytes).
     * s->lookahead stays null, so s->ins_h will be recomputed at the next
     * call of fill_window.
     */
    s->ins_h = s->window[0];
    UPDATE_HASH(s, s->ins_h, s->window[1]);
    for (n = 0; n <= length - MIN_MATCH; n++) {
    INSERT_STRING(s, n, hash_head);
    }
    if (hash_head) hash_head = 0;  /* to make compiler happy */
    return Z_OK;

/* ========================================================================= */
int ZEXPORT deflateReset (strm)
    z_streamp strm;

    deflate_state *s;

    if (strm == Z_NULL || strm->state == Z_NULL ||
        strm->zalloc == Z_NULL || strm->zfree == Z_NULL) return Z_STREAM_ERROR;

    strm->total_in = strm->total_out = 0;
    strm->msg = Z_NULL; /* use zfree if we ever allocate msg dynamically */
    strm->data_type = Z_UNKNOWN;

    s = (deflate_state *)strm->state;
    s->pending = 0;
    s->pending_out = s->pending_buf;

    if (s->noheader < 0) {
        s->noheader = 0; /* was set to -1 by deflate(..., Z_FINISH); */
    }
    s->status = s->noheader ? BUSY_STATE : INIT_STATE;
    strm->adler = 1;
    s->last_flush = Z_NO_FLUSH;

    _tr_init(s);
    lm_init(s);

    return Z_OK;
}

/* ========================================================================= */
int ZEXPORT deflateParams(strm, level, strategy)
    z_streamp strm;
    int level;
    int strategy;
```

```
     */

    if (version == Z_NULL || version[0] != my_version[0] ||
        stream_size != sizeof(z_stream)) {
    return Z_VERSION_ERROR;
    }
    if (strm == Z_NULL) return Z_STREAM_ERROR;

    strm->msg = Z_NULL;
    if (strm->zalloc == Z_NULL) {
    strm->zalloc = zcalloc;
    strm->opaque = (voidpf)0;
    }
    if (strm->zfree == Z_NULL) strm->zfree = zcfree;

    if (level == Z_DEFAULT_COMPRESSION) level = 6;
#ifdef FASTEST
    level = 1;
#endif

    if (windowBits < 0) { /* undocumented feature: suppress zlib header */
        noheader = 1;
        windowBits = -windowBits;
    }
    if (memLevel < 1 || memLevel > MAX_MEM_LEVEL || method != Z_DEFLATED ||
        windowBits < 8 || windowBits > 15 || level < 0 || level > 9 ||
    strategy < 0 || strategy > Z_HUFFMAN_ONLY) {
        return Z_STREAM_ERROR;
    }
    s = (deflate_state *) ZALLOC(strm, 1, sizeof(deflate_state));
    if (s == Z_NULL) return Z_MEM_ERROR;
    strm->state = (struct internal_state FAR *)s;
    s->strm = strm;

    s->noheader = noheader;
    s->w_bits = windowBits;
    s->w_size = 1 << s->w_bits;
    s->w_mask = s->w_size - 1;

    s->hash_bits = memLevel + 7;
    s->hash_size = 1 << s->hash_bits;
    s->hash_mask = s->hash_size - 1;
    s->hash_shift =  ((s->hash_bits+MIN_MATCH-1)/MIN_MATCH);

    s->window = (Bytef *) ZALLOC(strm, s->w_size, 2*sizeof(Byte));
    s->prev   = (Posf *)  ZALLOC(strm, s->w_size, sizeof(Pos));
    s->head   = (Posf *)  ZALLOC(strm, s->hash_size, sizeof(Pos));

    s->lit_bufsize = 1 << (memLevel + 6); /* 16K elements by default */

    overlay = (ushf *) ZALLOC(strm, s->lit_bufsize, sizeof(ush)+2);
    s->pending_buf = (uchf *) overlay;
    s->pending_buf_size = (ulg)s->lit_bufsize * (sizeof(ush)+2L);

    if (s->window == Z_NULL || s->prev == Z_NULL || s->head == Z_NULL ||
        s->pending_buf == Z_NULL) {
        strm->msg = (char*)ERR_MSG(Z_MEM_ERROR);
        deflateEnd (strm);
        return Z_MEM_ERROR;
    }
    s->d_buf = overlay + s->lit_bufsize/sizeof(ush);
    s->l_buf = s->pending_buf + (1+sizeof(ush))*s->lit_bufsize;

    s->level = level;
    s->strategy = strategy;
    s->method = (Byte)method;

    return deflateReset(strm);
}

/* ========================================================================= */
int ZEXPORT deflateSetDictionary (strm, dictionary, dictLength)
    z_streamp strm;
```

```
struct static_tree_desc_s {int dummy;}; /* for buggy compilers */

/* ===========================================================================
 * Update a hash value with the given input byte
 * IN  assertion: all calls to to UPDATE_HASH are made with consecutive
 *    input characters, so that a running hash key can be computed from the
 *    previous key instead of complete recalculation each time.
 */
#define UPDATE_HASH(s,h,c) (h = (((h)<<s->hash_shift) ^ (c)) & s->hash_mask)


/* ===========================================================================
 * Insert string str in the dictionary and set match_head to the previous head
 * of the hash chain (the most recent string with same hash key). Return
 * the previous length of the hash chain.
 * If this file is compiled with -DFASTEST, the compression level is forced
 * to 1, and no hash chains are maintained.
 * IN  assertion: all calls to to INSERT_STRING are made with consecutive
 *    input characters and the first MIN_MATCH bytes of str are valid
 *    (except for the last MIN_MATCH-1 bytes of the input file).
 */
#ifdef FASTEST
#define INSERT_STRING(s, str, match_head) \
   (UPDATE_HASH(s, s->ins_h, s->window[(str) + (MIN_MATCH-1)]), \
    match_head = s->head[s->ins_h], \
    s->head[s->ins_h] = (Pos)(str))
#else
#define INSERT_STRING(s, str, match_head) \
   (UPDATE_HASH(s, s->ins_h, s->window[(str) + (MIN_MATCH-1)]), \
    s->prev[(str) & s->w_mask] = match_head = s->head[s->ins_h], \
    s->head[s->ins_h] = (Pos)(str))
#endif

/* ===========================================================================
 * Initialize the hash table (avoiding 64K overflow for 16 bit systems).
 * prev[] will be initialized on the fly.
 */
#define CLEAR_HASH(s) \
    s->head[s->hash_size-1] = NIL; \
    zmemzero((Bytef *)s->head, (unsigned)(s->hash_size-1)*sizeof(*s->head));

/* ========================================================================= */
int ZEXPORT deflateInit_(strm, level, version, stream_size)
    z_streamp strm;
    int level;
    const char *version;
    int stream_size;
{
    return deflateInit2_(strm, level, Z_DEFLATED, MAX_WBITS, DEF_MEM_LEVEL,
                         Z_DEFAULT_STRATEGY, version, stream_size);
    /* To do: ignore strm->next_in if we use it as window */
}

/* ========================================================================= */
int ZEXPORT deflateInit2_(strm, level, method, windowBits, memLevel, strategy,
                  version, stream_size)
    z_streamp strm;
    int  level;
    int  method;
    int  windowBits;
    int  memLevel;
    int  strategy;
    const char *version;
    int stream_size;
{
    deflate_state *s;
    int noheader = 0;
    static const char* my_version = ZLIB_VERSION;

    ushf *overlay;
    /* We overlay pending_buf and d_buf+l_buf. This works since the average
     * output size for (length,distance) codes is <= 24 bits.
```

```
/* Compression function. Returns the block state after the call. */

local void fill_window    OF((deflate_state *s));
local block_state deflate_stored OF((deflate_state *s, int flush));
local block_state deflate_fast   OF((deflate_state *s, int flush));
local block_state deflate_slow   OF((deflate_state *s, int flush));
local void lm_init        OF((deflate_state *s));
local void putShortMSB    OF((deflate_state *s, uInt b));
local void flush_pending  OF((z_streamp strm));
local int read_buf        OF((z_streamp strm, Bytef *buf, unsigned size));
#ifdef ASMV
      void match_init OF((void)); /* asm code initialization */
      uInt longest_match  OF((deflate_state *s, IPos cur_match));
#else
local uInt longest_match  OF((deflate_state *s, IPos cur_match));
#endif

#ifdef DEBUG
local  void check_match OF((deflate_state *s, IPos start, IPos match,
                            int length));
#endif

/* ===========================================================================
 * Local data
 */

#define NIL 0
/* Tail of hash chains */

#ifndef TOO_FAR
#  define TOO_FAR 4096
#endif
/* Matches of length 3 are discarded if their distance exceeds TOO_FAR */

#define MIN_LOOKAHEAD (MAX_MATCH+MIN_MATCH+1)
/* Minimum amount of lookahead, except at the end of the input file.
 * See deflate.c for comments about the MIN_MATCH+1.
 */

/* Values for max_lazy_match, good_match and max_chain_length, depending on
 * the desired pack level (0..9). The values given below have been tuned to
 * exclude worst case performance for pathological files. Better values may be
 * found for specific files.
 */
typedef struct config_s {
   ush good_length; /* reduce lazy search above this match length */
   ush max_lazy;    /* do not perform lazy search above this match length */
   ush nice_length; /* quit search above this match length */
   ush max_chain;
   compress_func func;
} config;

local const config configuration_table[10] = {
/*      good lazy nice chain */
/* 0 */ {0,    0,  0,    0, deflate_stored},  /* store only */
/* 1 */ {4,    4,  8,    4, deflate_fast}, /* maximum speed, no lazy matches */
/* 2 */ {4,    5, 16,    8, deflate_fast},
/* 3 */ {4,    6, 32,   32, deflate_fast},

/* 4 */ {4,    4, 16,   16, deflate_slow},  /* lazy matches */
/* 5 */ {8,   16, 32,   32, deflate_slow},
/* 6 */ {8,   16, 128, 128, deflate_slow},
/* 7 */ {8,   32, 128, 256, deflate_slow},
/* 8 */ {32, 128, 258, 1024, deflate_slow},
/* 9 */ {32, 258, 258, 4096, deflate_slow}}; /* maximum compression */

/* Note: the deflate() code requires max_lazy >= MIN_MATCH and max_chain >= 4
 * For deflate_fast() (levels <= 3) good is ignored and lazy has a different
 * meaning.
 */

#define EQUAL 0
/* result of memcmp for equal strings */
/* deflate.c -- compress data using the deflation algorithm
 * Copyright (C) 1995-1998 Jean-loup Gailly.
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/*
 *  ALGORITHM
 *
 *      The "deflation" process depends on being able to identify portions
 *      of the input text which are identical to earlier input (within a
 *      sliding window trailing behind the input currently being processed).
 *
 *      The most straightforward technique turns out to be the fastest for
 *      most input files: try all possible matches and select the longest.
 *      The key feature of this algorithm is that insertions into the string
 *      dictionary are very simple and thus fast, and deletions are avoided
 *      completely. Insertions are performed at each input character, whereas
 *      string matches are performed only when the previous match ends. So it
 *      is preferable to spend more time in matches to allow very fast string
 *      insertions and avoid deletions. The matching algorithm for small
 *      strings is inspired from that of Rabin & Karp. A brute force approach
 *      is used to find longer strings when a small match has been found.
 *      A similar algorithm is used in comic (by Jan-Mark Wams) and freeze
 *      (by Leonid Broukhis).
 *         A previous version of this file used a more sophisticated algorithm
 *      (by Fiala and Greene) which is guaranteed to run in linear amortized
 *      time, but has a larger average cost, uses more memory and is patented.
 *      However the F&G algorithm may be faster for some highly redundant
 *      files if the parameter max_chain_length (described below) is too large.
 *
 *  ACKNOWLEDGEMENTS
 *
 *      The idea of lazy evaluation of matches is due to Jan-Mark Wams, and
 *      I found it in 'freeze' written by Leonid Broukhis.
 *      Thanks to many people for bug reports and testing.
 *
 *  REFERENCES
 *
 *      Deutsch, L.P.,"DEFLATE Compressed Data Format Specification".
 *      Available in ftp://ds.internic.net/rfc/rfc1951.txt
 *
 *      A description of the Rabin and Karp algorithm is given in the book
 *         "Algorithms" by R. Sedgewick, Addison-Wesley, p252.
 *
 *      Fiala,E.R., and Greene,D.H.
 *         Data Compression with Finite Windows, Comm.ACM, 32,4 (1989) 490-595
 *
 */

/* @(#) $Id$ */

#include "deflate.h"

const char deflate_copyright[] =
   " deflate 1.1.3 Copyright 1995-1998 Jean-loup Gailly ";
/*
  If you use the zlib library in a product, an acknowledgment is welcome
  in the documentation of your product. If for some reason you cannot
  include such an acknowledgment, I would appreciate that you keep this
  copyright string in the executable of your product.
 */

/* ===========================================================================
 *  Function prototypes.
 */
typedef enum {
    need_more,      /* block not completed, need more input or more output */
    block_done,     /* block flush performed */
    finish_started, /* finish started, need only more output at next deflate */
    finish_done     /* finish done, accept no more input or output */
} block_state;

typedef block_state (*compress_func) OF((deflate_state *s, int flush));
```

```
#endif

# define _tr_tally_lit(s, c, flush) \
  { uch cc = (c); \
    s->d_buf[s->last_lit] = 0; \
    s->l_buf[s->last_lit++] = cc; \
    s->dyn_ltree[cc].Freq++; \
    flush = (s->last_lit == s->lit_bufsize-1); \
   }
# define _tr_tally_dist(s, distance, length, flush) \
  { uch len = (length); \
    ush dist = (distance); \
    s->d_buf[s->last_lit] = dist; \
    s->l_buf[s->last_lit++] = len; \
    dist--; \
    s->dyn_ltree[_length_code[len]+LITERALS+1].Freq++; \
    s->dyn_dtree[d_code(dist)].Freq++; \
    flush = (s->last_lit == s->lit_bufsize-1); \
  }
#else
# define _tr_tally_lit(s, c, flush) flush = _tr_tally(s, 0, c)
# define _tr_tally_dist(s, distance, length, flush) \
              flush = _tr_tally(s, distance, length)
#endif

#endif
```

```
     *   - I can't count above 4
     */

    uInt last_lit;      /* running index in l_buf */

    ushf *d_buf;
    /* Buffer for distances. To simplify the code, d_buf and l_buf have
     * the same number of elements. To use different lengths, an extra flag
     * array would be necessary.
     */

    ulg opt_len;        /* bit length of current block with optimal trees */
    ulg static_len;     /* bit length of current block with static trees */
    uInt matches;       /* number of string matches in current block */
    int last_eob_len;   /* bit length of EOB code for last block */

#ifdef DEBUG
    ulg compressed_len; /* total bit length of compressed file mod 2^32 */
    ulg bits_sent;      /* bit length of compressed data sent mod 2^32 */
#endif

    ush bi_buf;
    /* Output buffer. bits are inserted starting at the bottom (least
     * significant bits).
     */
    int bi_valid;
    /* Number of valid bits in bi_buf.  All bits above the last valid bit
     * are always zero.
     */

} FAR deflate_state;

/* Output a byte on the stream.
 * IN assertion: there is enough room in pending_buf.
 */
#define put_byte(s, c) {s->pending_buf[s->pending++] = (c);}


#define MIN_LOOKAHEAD (MAX_MATCH+MIN_MATCH+1)
/* Minimum amount of lookahead, except at the end of the input file.
 * See deflate.c for comments about the MIN_MATCH+1.
 */

#define MAX_DIST(s)  ((s)->w_size-MIN_LOOKAHEAD)
/* In order to simplify the code, particularly on 16 bit machines, match
 * distances are limited to MAX_DIST instead of WSIZE.
 */

        /* in trees.c */
void _tr_init         OF((deflate_state *s));
int  _tr_tally        OF((deflate_state *s, unsigned dist, unsigned lc));
void _tr_flush_block  OF((deflate_state *s, charf *buf, ulg stored_len,
                int eof));
void _tr_align        OF((deflate_state *s));
void _tr_stored_block OF((deflate_state *s, charf *buf, ulg stored_len,
                          int eof));

#define d_code(dist) \
   ((dist) < 256 ? _dist_code[dist] : _dist_code[256+((dist)>>7)])
/* Mapping from a distance to a distance code. dist is the distance - 1 and
 * must not have side effects. _dist_code[256] and _dist_code[257] are never
 * used.
 */

#ifndef DEBUG
/* Inline versions of _tr_tally for speed: */

#if defined(GEN_TREES_H) || !defined(STDC)
  extern uch _length_code[];
  extern uch _dist_code[];
#else
  extern const uch _length_code[];
  extern const uch _dist_code[];
```

```
    uInt prev_length;
    /* Length of the best match at previous step. Matches not greater than this
     * are discarded. This is used in the lazy match evaluation.
     */

    uInt max_chain_length;
    /* To speed up deflation, hash chains are never searched beyond this
     * length.  A higher limit improves compression ratio but degrades the
     * speed.
     */

    uInt max_lazy_match;
    /* Attempt to find a better match only when the current match is strictly
     * smaller than this value. This mechanism is used only for compression
     * levels >= 4.
     */
#   define max_insert_length  max_lazy_match
    /* Insert new strings in the hash table only if the match length is not
     * greater than this length. This saves time but degrades compression.
     * max_insert_length is used only for compression levels <= 3.
     */

    int level;    /* compression level (1..9) */
    int strategy; /* favor or force Huffman coding*/

    uInt good_match;
    /* Use a faster search when the previous match is longer than this */

    int nice_match; /* Stop searching when current match exceeds this */

                /* used by trees.c: */
    /* Didn't use ct_data typedef below to supress compiler warning */
    struct ct_data_s dyn_ltree[HEAP_SIZE];   /* literal and length tree */
    struct ct_data_s dyn_dtree[2*D_CODES+1]; /* distance tree */
    struct ct_data_s bl_tree[2*BL_CODES+1];  /* Huffman tree for bit lengths */

    struct tree_desc_s l_desc;               /* desc. for literal tree */
    struct tree_desc_s d_desc;               /* desc. for distance tree */
    struct tree_desc_s bl_desc;              /* desc. for bit length tree */

    ush bl_count[MAX_BITS+1];
    /* number of codes at each bit length for an optimal tree */

    int heap[2*L_CODES+1];      /* heap used to build the Huffman trees */
    int heap_len;               /* number of elements in the heap */
    int heap_max;               /* element of largest frequency */
    /* The sons of heap[n] are heap[2*n] and heap[2*n+1]. heap[0] is not used.
     * The same heap array is used to build all trees.
     */

    uch depth[2*L_CODES+1];
    /* Depth of each subtree used as tie breaker for trees of equal frequency
     */

    uchf *l_buf;          /* buffer for literals or lengths */

    uInt  lit_bufsize;
    /* Size of match buffer for literals/lengths.  There are 4 reasons for
     * limiting lit_bufsize to 64K:
     *   - frequencies can be kept in 16 bit counters
     *   - if compression is not successful for the first block, all input
     *     data is still in the window so we can still emit a stored block even
     *     when input comes from standard input.  (This can also be done for
     *     all blocks if lit_bufsize is not greater than 32K.)
     *   - if compression is not successful for a file smaller than 64K, we can
     *     even emit a stored file instead of a stored block (saving 5 bytes).
     *     This is applicable only for zip (not gzip or zlib).
     *   - creating new Huffman trees less frequently may not provide fast
     *     adaptation to changes in the input data statistics. (Take for
     *     example a binary file with poorly compressible code followed by
     *     a highly compressible string table.) Smaller buffer sizes give
     *     fast adaptation but have of course the overhead of transmitting
     *     trees more frequently.
```

```
typedef ush Pos;
typedef Pos FAR Posf;
typedef unsigned IPos;

/* A Pos is an index in the character window. We use short instead of int to
 * save space in the various tables. IPos is used only for parameter passing.
 */

typedef struct internal_state {
    z_streamp strm;      /* pointer back to this zlib stream */
    int   status;        /* as the name implies */
    Bytef *pending_buf;  /* output still pending */
    ulg   pending_buf_size; /* size of pending_buf */
    Bytef *pending_out;  /* next pending byte to output to the stream */
    int   pending;       /* nb of bytes in the pending buffer */
    int   noheader;      /* suppress zlib header and adler32 */
    Byte  data_type;     /* UNKNOWN, BINARY or ASCII */
    Byte  method;        /* STORED (for zip only) or DEFLATED */
    int   last_flush;    /* value of flush param for previous deflate call */

                /* used by deflate.c: */

    uInt  w_size;        /* LZ77 window size (32K by default) */
    uInt  w_bits;        /* log2(w_size)  (8..16) */
    uInt  w_mask;        /* w_size - 1 */

    Bytef *window;
    /* Sliding window. Input bytes are read into the second half of the window,
     * and move to the first half later to keep a dictionary of at least wSize
     * bytes. With this organization, matches are limited to a distance of
     * wSize-MAX_MATCH bytes, but this ensures that IO is always
     * performed with a length multiple of the block size. Also, it limits
     * the window size to 64K, which is quite useful on MSDOS.
     * To do: use the user input buffer as sliding window.
     */

    ulg window_size;
    /* Actual size of window: 2*wSize, except when the user input buffer
     * is directly used as sliding window.
     */

    Posf *prev;
    /* Link to older string with same hash index. To limit the size of this
     * array to 64K, this link is maintained only for the last 32K strings.
     * An index in this array is thus a window index modulo 32K.
     */

    Posf *head; /* Heads of the hash chains or NIL. */

    uInt  ins_h;          /* hash index of string to be inserted */
    uInt  hash_size;      /* number of elements in hash table */
    uInt  hash_bits;      /* log2(hash_size) */
    uInt  hash_mask;      /* hash_size-1 */

    uInt  hash_shift;
    /* Number of bits by which ins_h must be shifted at each input
     * step. It must be such that after MIN_MATCH steps, the oldest
     * byte no longer takes part in the hash key, that is:
     *   hash_shift * MIN_MATCH >= hash_bits
     */

    long block_start;
    /* Window position at the beginning of the current output block. Gets
     * negative when the window is moved backwards.
     */

    uInt match_length;           /* length of best match */
    IPos prev_match;             /* previous match */
    int match_available;         /* set if previous match exists */
    uInt strstart;               /* start of string to insert */
    uInt match_start;            /* start of matching string */
    uInt lookahead;              /* number of valid bytes ahead in window */
```

```
/* deflate.h -- internal compression state
 * Copyright (C) 1995-1998 Jean-loup Gailly
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* WARNING: this file should *not* be used by applications. It is
   part of the implementation of the compression library and is
   subject to change. Applications should only use zlib.h.
 */

/* @(#) $Id$ */

#ifndef _DEFLATE_H
#define _DEFLATE_H

#include "zutil.h"

/* ===========================================================================
 * Internal compression state.
 */

#define LENGTH_CODES 29
/* number of length codes, not counting the special END_BLOCK code */

#define LITERALS  256
/* number of literal bytes 0..255 */

#define L_CODES (LITERALS+1+LENGTH_CODES)
/* number of Literal or Length codes, including the END_BLOCK code */

#define D_CODES   30
/* number of distance codes */

#define BL_CODES  19
/* number of codes used to transfer the bit lengths */

#define HEAP_SIZE (2*L_CODES+1)
/* maximum heap size */

#define MAX_BITS 15
/* All codes must not exceed MAX_BITS bits */

#define INIT_STATE    42
#define BUSY_STATE   113
#define FINISH_STATE 666
/* Stream status */


/* Data structure describing a single value and its code string. */
typedef struct ct_data_s {
    union {
        ush  freq;       /* frequency count */
        ush  code;       /* bit string */
    } fc;
    union {
        ush  dad;        /* father node in Huffman tree */
        ush  len;        /* length of bit string */
    } dl;
} FAR ct_data;

#define Freq fc.freq
#define Code fc.code
#define Dad  dl.dad
#define Len  dl.len

typedef struct static_tree_desc_s  static_tree_desc;

typedef struct tree_desc_s {
    ct_data *dyn_tree;           /* the dynamic tree */
    int     max_code;            /* largest code with non zero frequency */
    static_tree_desc *stat_desc; /* the corresponding static tree */
} FAR tree_desc;
```

```
    int *errnum;
{
    char *m;
    gz_stream *s = (gz_stream*)file;

    if (s == NULL) {
        *errnum = Z_STREAM_ERROR;
        return (const char*)ERR_MSG(Z_STREAM_ERROR);
    }
    *errnum = s->z_err;
    if (*errnum == Z_OK) return (const char*)"";

    m =  (char*)(*errnum == Z_ERRNO ? zstrerror(errno) : s->stream
.msg);

    if (m == NULL || *m == '\0') m = (char*)ERR_MSG(s->z_err);

    TRYFREE(s->msg);
    s->msg = (char*)ALLOC(strlen(s->path) + strlen(m) + 3);
    strcpy(s->msg, s->path);
    strcat(s->msg, ": ");
    strcat(s->msg, m);
    return (const char*)s->msg;
}
```

```
    if (c == EOF) s->z_err = Z_DATA_ERROR;
    x += ((uLong)c)<<24;
    return x;
}

/* ===========================================================================
     Flushes all pending output if necessary, closes the compresse
d file
   and deallocates all the (de)compression state.
*/
int ZEXPORT gzclose (file)
    gzFile file;
{
    int err;
    gz_stream *s = (gz_stream*)file;

    if (s == NULL) return Z_STREAM_ERROR;

    if (s->mode == 'w') {
#ifdef NO_DEFLATE
        return Z_STREAM_ERROR;
#else
        err = do_flush (file, Z_FINISH);
        if (err != Z_OK) return destroy((gz_stream*)file);

        putLong (s->file, s->crc);
        putLong (s->file, s->stream.total_in);
#endif
    }
    return destroy((gz_stream*)file);
}

/* ===========================================================================
     Returns the error message for the last error which occured on
 the
   given compressed file. errnum is set to zlib error number. If a
n
   error occured in the file system and not in the compression lib
rary,
   errnum is set to Z_ERRNO and the application may consult errno
   to get the exact error code.
*/
const char*  ZEXPORT gzerror (file, errnum)
    gzFile file;
```

```
}

/* ===========================================================================
     Returns 1 when EOF has previously been detected reading the g
iven
   input stream, otherwise zero.
*/
int ZEXPORT gzeof (file)
    gzFile file;
{
    gz_stream *s = (gz_stream*)file;

    return (s == NULL || s->mode != 'r') ? 0 : s->z_eof;
}

/* ===========================================================================
   Outputs a long in LSB order to the given file
*/
local void putLong (file, x)
    FILE *file;
    uLong x;
{
    int n;
    for (n = 0; n < 4; n++) {
        fputc((int)(x & 0xff), file);
        x >>= 8;
    }
}

/* ===========================================================================
   Reads a long in LSB order from the given gz_stream. Sets z_err
in case
   of error.
*/
local uLong getLong (s)
    gz_stream *s;
{
    uLong x = (uLong)get_byte(s);
    int c;

    x += ((uLong)get_byte(s))<<8;
    x += ((uLong)get_byte(s))<<16;
    c = get_byte(s);
```

```
        size = gzread(file, s->outbuf, (uInt)size);
        if (size <= 0) return -1L;
        offset -= size;
    }
    return (z_off_t)s->stream.total_out;
}

/* ==================================================================
============
     Rewinds input file.
*/
int ZEXPORT gzrewind (file)
    gzFile file;
{
    gz_stream *s = (gz_stream*)file;

    if (s == NULL || s->mode != 'r') return -1;

    s->z_err = Z_OK;
    s->z_eof = 0;
    s->stream.avail_in = 0;
    s->stream.next_in = s->inbuf;
    s->crc = crc32(0L, Z_NULL, 0);

    if (s->startpos == 0) { /* not a compressed file */
        rewind(s->file);
        return 0;
    }

    (void) inflateReset(&s->stream);
    return fseek(s->file, s->startpos, SEEK_SET);
}

/* ==================================================================
============
     Returns the starting position for the next gzread or gzwrite
on the
   given compressed file. This position represents a number of byt
es in the
   uncompressed data stream.
*/
z_off_t ZEXPORT gztell (file)
    gzFile file;
{
    return gzseek(file, 0L, SEEK_CUR);
```

```
            zmemzero(s->inbuf, Z_BUFSIZE);
        }
        while (offset > 0)  {
            uInt size = Z_BUFSIZE;
            if (offset < Z_BUFSIZE) size = (uInt)offset;

            size = gzwrite(file, s->inbuf, size);
            if (size == 0) return -1L;

            offset -= size;
        }
        return (z_off_t)s->stream.total_in;
#endif
    }
    /* Rest of function is for reading only */

    /* compute absolute position */
    if (whence == SEEK_CUR) {
        offset += s->stream.total_out;
    }
    if (offset < 0) return -1L;

    if (s->transparent) {
        /* map to fseek */
        s->stream.avail_in = 0;
        s->stream.next_in = s->inbuf;
        if (fseek(s->file, offset, SEEK_SET) < 0) return -1L;

        s->stream.total_in = s->stream.total_out = (uLong)offset;
        return offset;
    }

    /* For a negative seek, rewind and use positive seek */
    if ((uLong)offset >= s->stream.total_out) {
        offset -= s->stream.total_out;
    } else if (gzrewind(file) < 0) {
        return -1L;
    }
    /* offset is now the number of bytes to skip. */

    if (offset != 0 && s->outbuf == Z_NULL) {
        s->outbuf = (Byte*)ALLOC(Z_BUFSIZE);
    }
    while (offset > 0)  {
        int size = Z_BUFSIZE;
        if (offset < Z_BUFSIZE) size = (int)offset;
    int err = do_flush (file, flush);

    if (err) return err;
    fflush(s->file);
    return  s->z_err == Z_STREAM_END ? Z_OK : s->z_err;
}
#endif /* NO_DEFLATE */

/* ===================================================================
============
      Sets the starting position for the next gzread or gzwrite on
 the given
   compressed file. The offset represents a number of bytes in the
      gzseek returns the resulting offset location as measured in
bytes from
   the beginning of the uncompressed stream, or -1 in case of erro
r.
      SEEK_END is not implemented, returns error.
      In this version of the library, gzseek can be extremely slow
.
*/
z_off_t ZEXPORT gzseek (file, offset, whence)
    gzFile file;
    z_off_t offset;
    int whence;
{
    gz_stream *s = (gz_stream*)file;

    if (s == NULL || whence == SEEK_END ||
        s->z_err == Z_ERRNO || s->z_err == Z_DATA_ERROR) {
        return -1L;
    }

    if (s->mode == 'w') {
#ifdef NO_DEFLATE
        return -1L;
#else
        if (whence == SEEK_SET) {
            offset -= s->stream.total_in;
        }
        if (offset < 0) return -1L;

        /* At this point, offset is the number of zero bytes to wr
ite. */
        if (s->inbuf == Z_NULL) {
            s->inbuf = (Byte*)ALLOC(Z_BUFSIZE); /* for seeking */
```

```
local int do_flush (file, flush)
    gzFile file;
    int flush;
{
    uInt len;
    int done = 0;
    gz_stream *s = (gz_stream*)file;

    if (s == NULL || s->mode != 'w') return Z_STREAM_ERROR;

    s->stream.avail_in = 0; /* should be zero already anyway */

    for (;;) {
        len = Z_BUFSIZE - s->stream.avail_out;

        if (len != 0) {
            if ((uInt)fwrite(s->outbuf, 1, len, s->file) != len) {
                s->z_err = Z_ERRNO;
                return Z_ERRNO;
            }
            s->stream.next_out = s->outbuf;
            s->stream.avail_out = Z_BUFSIZE;
        }
        if (done) break;
        s->z_err = deflate(&(s->stream), flush);

        /* Ignore the second of two consecutive flushes: */
        if (len == 0 && s->z_err == Z_BUF_ERROR) s->z_err = Z_OK;

        /* deflate has finished flushing only when it hasn't used
up
         * all the available space in the output buffer:
         */
        done = (s->stream.avail_out != 0 || s->z_err == Z_STREAM_E
ND);

        if (s->z_err != Z_OK && s->z_err != Z_STREAM_END) break;
    }
    return  s->z_err == Z_STREAM_END ? Z_OK : s->z_err;
}

int ZEXPORT gzflush (file, flush)
     gzFile file;
     int flush;
{
    gz_stream *s = (gz_stream*)file;
    if (len <= 0) return 0;

    return gzwrite(file, buf, len);
}
#endif

/* ===================================================================
============
      Writes c, converted to an unsigned char, into the compressed
 file.
   gzputc returns the value that was written, or -1 in case of err
or.
*/
int ZEXPORT gzputc(file, c)
    gzFile file;
    int c;
{
    unsigned char cc = (unsigned char) c; /* required for big endi
an systems */

    return gzwrite(file, &cc, 1) == 1 ? (int)cc : -1;
}


/* ===================================================================
============
      Writes the given null-terminated string to the compressed fi
le, excluding
   the terminating null character.
      gzputs returns the number of characters written, or -1 in ca
se of error.
*/
int ZEXPORT gzputs(file, s)
    gzFile file;
    const char *s;
{
    return gzwrite(file, (char*)s, (unsigned)strlen(s));
}


/* ===================================================================
============
     Flushes all pending output into the compressed file. The para
meter
   flush is as in the deflate() function.
*/
```

```
int ZEXPORTVA gzprintf (gzFile file, const char *format, /* args *
/ ...)
{
    char buf[Z_PRINTF_BUFSIZE];
    va_list va;
    int len;

    va_start(va, format);
#ifdef HAS_vsnprintf
    (void)vsnprintf(buf, sizeof(buf), format, va);
#else
    (void)vsprintf(buf, format, va);
#endif
    va_end(va);
    len = strlen(buf); /* some *sprintf don't return the nb of byt
es written */
    if (len <= 0) return 0;

    return gzwrite(file, buf, (unsigned)len);
}
#else /* not ANSI C */

int ZEXPORTVA gzprintf (file, format, a1, a2, a3, a4, a5, a6, a7,
a8, a9, a10,
                       a11, a12, a13, a14, a15, a16, a17, a18, a19
, a20)
    gzFile file;
    const char *format;
    int a1, a2, a3, a4, a5, a6, a7, a8, a9, a10,
        a11, a12, a13, a14, a15, a16, a17, a18, a19, a20;
{
    char buf[Z_PRINTF_BUFSIZE];
    int len;

#ifdef HAS_snprintf
    snprintf(buf, sizeof(buf), format, a1, a2, a3, a4, a5, a6, a7,
 a8,
             a9, a10, a11, a12, a13, a14, a15, a16, a17, a18, a19,
 a20);
#else
    sprintf(buf, format, a1, a2, a3, a4, a5, a6, a7, a8,
            a9, a10, a11, a12, a13, a14, a15, a16, a17, a18, a19,
a20);
#endif
    len = strlen(buf); /* old sprintf doesn't return the nb of byt
es written */
```

```
   gzwrite returns the number of bytes actually written (0 in case
 of error).
*/
int ZEXPORT gzwrite (file, buf, len)
    gzFile file;
    const voidp buf;
    unsigned len;
{
    gz_stream *s = (gz_stream*)file;

    if (s == NULL || s->mode != 'w') return Z_STREAM_ERROR;

    s->stream.next_in = (Bytef*)buf;
    s->stream.avail_in = len;

    while (s->stream.avail_in != 0) {

        if (s->stream.avail_out == 0) {

            s->stream.next_out = s->outbuf;
            if (fwrite(s->outbuf, 1, Z_BUFSIZE, s->file) != Z_BUFS
IZE) {
                s->z_err = Z_ERRNO;
                break;
            }
            s->stream.avail_out = Z_BUFSIZE;
        }
        s->z_err = deflate(&(s->stream), Z_NO_FLUSH);
        if (s->z_err != Z_OK) break;
    }
    s->crc = crc32(s->crc, (const Bytef *)buf, len);

    return (int)(len - s->stream.avail_in);
}

/* ===================================================================
============
     Converts, formats, and writes the args to the compressed file
 under
   control of the format string, as in fprintf. gzprintf returns t
he number of
   uncompressed bytes actually written (0 in case of error).
*/
#ifdef STDC
#include <stdarg.h>
```

```
      Reads one byte from the compressed file. gzgetc returns this
 byte
   or -1 in case of end of file or error.
*/
int ZEXPORT gzgetc(file)
    gzFile file;
{
    unsigned char c;

    return gzread(file, &c, 1) == 1 ? c : -1;
}


/* ===========================================================
============
      Reads bytes from the compressed file until len-1 characters
are
   read, or a newline character is read and transferred to buf, or
 an
   end-of-file condition is encountered.  The string is then termi
nated
   with a null character.
      gzgets returns buf, or Z_NULL in case of error.

      The current implementation is not optimized at all.
*/
char * ZEXPORT gzgets(file, buf, len)
    gzFile file;
    char *buf;
    int len;
{
    char *b = buf;
    if (buf == Z_NULL || len <= 0) return Z_NULL;

    while (--len > 0 && gzread(file, buf, 1) == 1 && *buf++ != '\n
') ;
    *buf = '\0';
    return b == buf && len > 0 ? Z_NULL : b;
}


#ifndef NO_DEFLATE
/* ===========================================================
============
     Writes the given number of uncompressed bytes into the compre
ssed file.
```

```
                    break;
                }
            }
            s->stream.next_in = s->inbuf;
        }
        s->z_err = inflate(&(s->stream), Z_NO_FLUSH);

        if (s->z_err == Z_STREAM_END) {
            /* Check CRC and original size */
            s->crc = crc32(s->crc, start, (uInt)(s->stream.next_ou
t - start));
            start = s->stream.next_out;

            if (getLong(s) != s->crc) {
                s->z_err = Z_DATA_ERROR;
            } else {
                (void)getLong(s);
                /* The uncompressed length returned by above getlo
ng() may
                 * be different from s->stream.total_out) in case
of
                 * concatenated .gz files. Check for such files:
                 */
                check_header(s);
                if (s->z_err == Z_OK) {
                    uLong total_in = s->stream.total_in;
                    uLong total_out = s->stream.total_out;

                    inflateReset(&(s->stream));
                    s->stream.total_in = total_in;
                    s->stream.total_out = total_out;
                    s->crc = crc32(0L, Z_NULL, 0);
                }
            }
        }
        if (s->z_err != Z_OK || s->z_eof) break;
    }
    s->crc = crc32(s->crc, start, (uInt)(s->stream.next_out - star
t));

    return (int)(len - s->stream.avail_out);
}


/* ===================================================================
============
 MSDOS) */

    if (s == NULL || s->mode != 'r') return Z_STREAM_ERROR;

    if (s->z_err == Z_DATA_ERROR || s->z_err == Z_ERRNO) return -1
;
    if (s->z_err == Z_STREAM_END) return 0;  /* EOF */

    next_out = (Byte*)buf;
    s->stream.next_out = (Bytef*)buf;
    s->stream.avail_out = len;

    while (s->stream.avail_out != 0) {

        if (s->transparent) {
            /* Copy first the lookahead bytes: */
            uInt n = s->stream.avail_in;
            if (n > s->stream.avail_out) n = s->stream.avail_out;
            if (n > 0) {
                zmemcpy(s->stream.next_out, s->stream.next_in, n);
                next_out += n;
                s->stream.next_out = next_out;
                s->stream.next_in   += n;
                s->stream.avail_out -= n;
                s->stream.avail_in  -= n;
            }
            if (s->stream.avail_out > 0) {
                s->stream.avail_out -= fread(next_out, 1, s->strea
m.avail_out,
                                             s->file);
            }
            len -= s->stream.avail_out;
            s->stream.total_in  += (uLong)len;
            s->stream.total_out += (uLong)len;
            if (len == 0) s->z_eof = 1;
            return (int)len;
        }
        if (s->stream.avail_in == 0 && !s->z_eof) {

            errno = 0;
            s->stream.avail_in = fread(s->inbuf, 1, Z_BUFSIZE, s->
file);
            if (s->stream.avail_in == 0) {
                s->z_eof = 1;
                if (ferror(s->file)) {
                    s->z_err = Z_ERRNO;
    TRYFREE(s->msg);

    if (s->stream.state != NULL) {
        if (s->mode == 'w') {
#ifdef NO_DEFLATE
            err = Z_STREAM_ERROR;
#else
            err = deflateEnd(&(s->stream));
#endif
        } else if (s->mode == 'r') {
            err = inflateEnd(&(s->stream));
        }
    }
    if (s->file != NULL && fclose(s->file)) {
#ifdef ESPIPE
        if (errno != ESPIPE) /* fclose is broken for pipes in HP/U
X */
#endif
            err = Z_ERRNO;
    }
    if (s->z_err < 0) err = s->z_err;

    TRYFREE(s->inbuf);
    TRYFREE(s->outbuf);
    TRYFREE(s->path);
    TRYFREE(s);
    return err;
}

/* ===================================================================
============
     Reads the given number of uncompressed bytes from the compres
sed file.
   gzread returns the number of bytes actually read (0 for end of
file).
*/
int ZEXPORT gzread (file, buf, len)
    gzFile file;
    voidp buf;
    unsigned len;
{
    gz_stream *s = (gz_stream*)file;
    Bytef *start = (Bytef*)buf; /* starting point for crc computat
ion */
    Byte  *next_out; /* == stream.next_out but not forced far (for
                return;
            }
        }
        method = get_byte(s);
        flags = get_byte(s);
        if (method != Z_DEFLATED || (flags & RESERVED) != 0) {
            s->z_err = Z_DATA_ERROR;
            return;
        }

        /* Discard time, xflags and OS code: */
        for (len = 0; len < 6; len++) (void)get_byte(s);

        if ((flags & EXTRA_FIELD) != 0) { /* skip the extra field */
            len  =  (uInt)get_byte(s);
            len += ((uInt)get_byte(s))<<8;
            /* len is garbage if EOF but the loop below will quit anyw
ay */
            while (len-- != 0 && get_byte(s) != EOF) ;
        }
        if ((flags & ORIG_NAME) != 0) { /* skip the original file name
 */
            while ((c = get_byte(s)) != 0 && c != EOF) ;
        }
        if ((flags & COMMENT) != 0) {   /* skip the .gz file comment *
/
            while ((c = get_byte(s)) != 0 && c != EOF) ;
        }
        if ((flags & HEAD_CRC) != 0) {  /* skip the header crc */
            for (len = 0; len < 2; len++) (void)get_byte(s);
        }
        s->z_err = s->z_eof ? Z_DATA_ERROR : Z_OK;
    }

 /* ===========================================================
=============
 * Cleanup then free the given gz_stream. Return a zlib error code
.
   Try freeing in the reverse order of allocations.
 */
local int destroy (s)
    gz_stream *s;
{
    int err = Z_OK;

    if (!s) return Z_STREAM_ERROR;
```

```
        if (s->stream.avail_in == 0) {
            s->z_eof = 1;
            if (ferror(s->file)) s->z_err = Z_ERRNO;
            return EOF;
        }
        s->stream.next_in = s->inbuf;
    }
    s->stream.avail_in--;
    return *(s->stream.next_in)++;
}

/* ================================================================
============
      Check the gzip header of a gz_stream opened for reading. Set
 the stream
    mode to transparent if the gzip magic header is not present; s
et s->err
    to Z_DATA_ERROR if the magic header is present but the rest of
 the header
    is incorrect.
    IN assertion: the stream s has already been created sucessfull
y;
       s->stream.avail_in is zero for the first time, but may be n
on-zero
       for concatenated .gz files.
*/
local void check_header(s)
    gz_stream *s;
{
    int method; /* method byte */
    int flags;  /* flags byte */
    uInt len;
    int c;

    /* Check the gzip magic header */
    for (len = 0; len < 2; len++) {
        c = get_byte(s);
        if (c != gz_magic[len]) {
            if (len != 0) s->stream.avail_in++, s->stream.next_in-
-;
            if (c != EOF) {
                s->stream.avail_in++, s->stream.next_in--;
                s->transparent = 1;
            }
            s->z_err = s->stream.avail_in != 0 ? Z_OK : Z_STREAM_E
ND;
```

```
    return gz_open (name, mode, fd);
}

/* ===============================================================
============
 * Update the compression level and strategy
 */
int ZEXPORT gzsetparams (file, level, strategy)
    gzFile file;
    int level;
    int strategy;
{
    gz_stream *s = (gz_stream*)file;

    if (s == NULL || s->mode != 'w') return Z_STREAM_ERROR;

    /* Make room to allow flushing */
    if (s->stream.avail_out == 0) {

        s->stream.next_out = s->outbuf;
        if (fwrite(s->outbuf, 1, Z_BUFSIZE, s->file) != Z_BUFSIZE)
 {
            s->z_err = Z_ERRNO;
        }
        s->stream.avail_out = Z_BUFSIZE;
    }

    return deflateParams (&(s->stream), level, strategy);
}

/* ===============================================================
============
     Read a byte from a gz_stream; update next_in and avail_in. Re
turn EOF
   for end of file.
   IN assertion: the stream s has been sucessfully opened for read
ing.
*/
local int get_byte(s)
    gz_stream *s;
{
    if (s->z_eof) return EOF;
    if (s->stream.avail_in == 0) {
        errno = 0;
        s->stream.avail_in = fread(s->inbuf, 1, Z_BUFSIZE, s->file
);
agic[1],
             Z_DEFLATED, 0 /*flags*/, 0,0,0,0 /*time*/, 0 /*xflags
*/, OS_CODE);
        s->startpos = 10L;
        /* We use 10L instead of ftell(s->file) to because ftell c
auses an
         * fflush on some systems. This version of the library doe
sn't use
         * startpos anyway in write mode, so this initialization i
s not
         * necessary.
         */
    } else {
        check_header(s); /* skip the .gz header */
        s->startpos = (ftell(s->file) - s->stream.avail_in);
    }

    return (gzFile)s;
}

/* ===============================================================
============
     Opens a gzip (.gz) file for reading or writing.
*/
gzFile ZEXPORT gzopen (path, mode)
    const char *path;
    const char *mode;
{
    return gz_open (path, mode, -1);
}

/* ===============================================================
============
     Associate a gzFile with the file descriptor fd. fd is not dup
'ed here
   to mimic the behavio(u)r of fdopen.
*/
gzFile ZEXPORT gzdopen (fd, mode)
    int fd;
    const char *mode;
{
    char name[20];

    if (fd < 0) return (gzFile)Z_NULL;
    sprintf(name, "<fd:%d>", fd); /* for debugging */
    if (s->mode == 'w') {
#ifdef NO_DEFLATE
        err = Z_STREAM_ERROR;
#else
        err = deflateInit2(&(s->stream), level,
                           Z_DEFLATED, -MAX_WBITS, DEF_MEM_LEVEL,
strategy);
        /* windowBits is passed < 0 to suppress zlib header */

        s->stream.next_out = s->outbuf = (Byte*)ALLOC(Z_BUFSIZE);
#endif
        if (err != Z_OK || s->outbuf == Z_NULL) {
            return destroy(s), (gzFile)Z_NULL;
        }
    } else {
        s->stream.next_in  = s->inbuf = (Byte*)ALLOC(Z_BUFSIZE);

        err = inflateInit2(&(s->stream), -MAX_WBITS);
        /* windowBits is passed < 0 to tell that there is no zlib
header.
         * Note that in this case inflate *requires* an extra "dum
my" byte
         * after the compressed stream in order to complete decomp
ression and
         * return Z_STREAM_END. Here the gzip CRC32 ensures that 4
 bytes are
         * present after the compressed stream.
         */
        if (err != Z_OK || s->inbuf == Z_NULL) {
            return destroy(s), (gzFile)Z_NULL;
        }
    }
    s->stream.avail_out = Z_BUFSIZE;

    errno = 0;
    s->file = fd < 0 ? F_OPEN(path, fmode) : (FILE*)fdopen(fd, fmo
de);

    if (s->file == NULL) {
        return destroy(s), (gzFile)Z_NULL;
    }
    if (s->mode == 'w') {
        /* Write a very simple .gz header:
         */
        fprintf(s->file, "%c%c%c%c%c%c%c%c%c%c", gz_magic[0], gz_m
    int strategy = Z_DEFAULT_STRATEGY; /* compression strategy */
    char *p = (char*)mode;
    gz_stream *s;
    char fmode[80]; /* copy of mode, without the compression level
 */
    char *m = fmode;

    if (!path || !mode) return Z_NULL;

    s = (gz_stream *)ALLOC(sizeof(gz_stream));
    if (!s) return Z_NULL;

    s->stream.zalloc = (alloc_func)0;
    s->stream.zfree = (free_func)0;
    s->stream.opaque = (voidpf)0;
    s->stream.next_in = s->inbuf = Z_NULL;
    s->stream.next_out = s->outbuf = Z_NULL;
    s->stream.avail_in = s->stream.avail_out = 0;
    s->file = NULL;
    s->z_err = Z_OK;
    s->z_eof = 0;
    s->crc = crc32(0L, Z_NULL, 0);
    s->msg = NULL;
    s->transparent = 0;

    s->path = (char*)ALLOC(strlen(path)+1);
    if (s->path == NULL) {
        return destroy(s), (gzFile)Z_NULL;
    }
    strcpy(s->path, path); /* do this early for debugging */

    s->mode = '\0';
    do {
        if (*p == 'r') s->mode = 'r';
        if (*p == 'w' || *p == 'a') s->mode = 'w';
        if (*p >= '0' && *p <= '9') {
            level = *p - '0';
        } else if (*p == 'f') {
          strategy = Z_FILTERED;
        } else if (*p == 'h') {
          strategy = Z_HUFFMAN_ONLY;
        } else {
            *m++ = *p; /* copy the mode */
        }
    } while (*p++ && m != fmode + sizeof(fmode));
    if (s->mode == '\0') return destroy(s), (gzFile)Z_NULL;
    int      z_err;   /* error code for last stream operation */
    int      z_eof;   /* set if end of input file */
    FILE     *file;   /* .gz file */
    Byte     *inbuf;  /* input buffer */
    Byte     *outbuf; /* output buffer */
    uLong    crc;     /* crc32 of uncompressed data */
    char     *msg;    /* error message */
    char     *path;   /* path name for debugging only */
    int      transparent; /* 1 if input file is not a .gz file */
    char     mode;    /* 'w' or 'r' */
    long     startpos; /* start of compressed data in file (header
 skipped) */
} gz_stream;


local gzFile gz_open      OF((const char *path, const char *mode,
int  fd));
local int do_flush        OF((gzFile file, int flush));
local int    get_byte     OF((gz_stream *s));
local void   check_header OF((gz_stream *s));
local int    destroy      OF((gz_stream *s));
local void   putLong      OF((FILE *file, uLong x));
local uLong  getLong      OF((gz_stream *s));

/* ===================================================================
============
     Opens a gzip (.gz) file for reading or writing. The mode para
meter
   is as in fopen ("rb" or "wb"). The file is given either by file
 descriptor
   or path name (if fd == -1).
     gz_open return NULL if the file could not be opened or if the
re was
   insufficient memory to allocate the (de)compression state; errn
o
   can be checked to distinguish the two cases (if errno is zero,
the
   zlib error is Z_MEM_ERROR).
*/
local gzFile gz_open (path, mode, fd)
    const char *path;
    const char *mode;
    int  fd;
{
    int err;
    int level = Z_DEFAULT_COMPRESSION; /* compression level */
```

```
/* gzio.c -- IO on .gz files
 * Copyright (C) 1995-1998 Jean-loup Gailly.
 * For conditions of distribution and use, see copyright notice in
 zlib.h
 *
 * Compile this file with -DNO_DEFLATE to avoid the compression co
de.
 */

/* @(#) $Id$ */

#include <stdio.h>

#include "zutil.h"

struct internal_state {int dummy;}; /* for buggy compilers */

#ifndef Z_BUFSIZE
#  ifdef MAXSEG_64K
#    define Z_BUFSIZE 4096 /* minimize memory usage for 16-bit DOS
 */
#  else
#    define Z_BUFSIZE 16384
#  endif
#endif
#ifndef Z_PRINTF_BUFSIZE
#  define Z_PRINTF_BUFSIZE 4096
#endif

#define ALLOC(size) malloc(size)
#define TRYFREE(p) {if (p) free(p);}

static int gz_magic[2] = {0x1f, 0x8b}; /* gzip magic header */

/* gzip flag byte */
#define ASCII_FLAG   0x01 /* bit 0 set: file probably ascii text *
/
#define HEAD_CRC     0x02 /* bit 1 set: header CRC present */
#define EXTRA_FIELD  0x04 /* bit 2 set: extra field present */
#define ORIG_NAME    0x08 /* bit 3 set: original file name present
 */
#define COMMENT      0x10 /* bit 4 set: file comment present */
#define RESERVED     0xE0 /* bits 5..7: reserved */

typedef struct gz_stream {
    z_stream stream;
```

```
int inflate_blocks_free(s, z)
inflate_blocks_statef *s;
z_streamp z;
{
  inflate_blocks_reset(s, z, Z_NULL);
  ZFREE(z, s->window);
  ZFREE(z, s->hufts);
  ZFREE(z, s);
  Tracev((stderr, "inflate:   blocks freed\n"));
  return Z_OK;
}


void inflate_set_dictionary(s, d, n)
inflate_blocks_statef *s;
const Bytef *d;
uInt  n;
{
  zmemcpy(s->window, d, n);
  s->read = s->write = s->window + n;
}


/* Returns true if inflate is currently at the end of a block generated
 * by Z_SYNC_FLUSH or Z_FULL_FLUSH.
 * IN assertion: s != Z_NULL
 */
int inflate_blocks_sync_point(s)
inflate_blocks_statef *s;
{
  return s->mode == LENS;
}
```

```
            z->msg = (char*)"invalid bit length repeat";
            r = Z_DATA_ERROR;
            LEAVE
          }
          c = c == 16 ? s->sub.trees.blens[i - 1] : 0;
          do {
            s->sub.trees.blens[i++] = c;
          } while (--j);
          s->sub.trees.index = i;
        }
      }
      s->sub.trees.tb = Z_NULL;
      {
        uInt bl, bd;
        inflate_huft *tl, *td;
        inflate_codes_statef *c;

        bl = 9;         /* must be <= 9 for lookahead assumptions */
        bd = 6;         /* must be <= 9 for lookahead assumptions */
        t = s->sub.trees.table;
        t = inflate_trees_dynamic(257 + (t & 0x1f), 1 + ((t >> 5) & 0x1f),
                                  s->sub.trees.blens, &bl, &bd, &tl, &td,
                                  s->hufts, z);
        ZFREE(z, s->sub.trees.blens);
        if (t != Z_OK)
        {
          if (t == (uInt)Z_DATA_ERROR)
            s->mode = BAD;
          r = t;
          LEAVE
        }
        Tracev((stderr, "inflate:       trees ok\n"));
        if ((c = inflate_codes_new(bl, bd, tl, td, z)) == Z_NULL)
        {
          r = Z_MEM_ERROR;
          LEAVE
        }
        s->sub.decode.codes = c;
      }
      s->mode = CODES;
    case CODES:
      UPDATE
      if ((r = inflate_codes(s, z, r)) != Z_STREAM_END)
        return inflate_flush(s, z, r);
      r = Z_OK;
      inflate_codes_free(s->sub.decode.codes, z);
      LOAD
      Tracev((stderr, "inflate:       codes end, %lu total out\n",
              z->total_out + (q >= s->read ? q - s->read :
              (s->end - s->read) + (q - s->window))));
      if (!s->last)
      {
        s->mode = TYPE;
        break;
      }
      s->mode = DRY;
    case DRY:
      FLUSH
      if (s->read != s->write)
        LEAVE
      s->mode = DONE;
    case DONE:
      r = Z_STREAM_END;
      LEAVE
    case BAD:
      r = Z_DATA_ERROR;
      LEAVE
    default:
      r = Z_STREAM_ERROR;
      LEAVE
  }
}
```

```
      if ((t & 0x1f) > 29 || ((t >> 5) & 0x1f) > 29)
      {
        s->mode = BAD;
        z->msg = (char*)"too many length or distance symbols";
        r = Z_DATA_ERROR;
        LEAVE
      }
#endif
      t = 258 + (t & 0x1f) + ((t >> 5) & 0x1f);
      if ((s->sub.trees.blens = (uIntf*)ZALLOC(z, t, sizeof(uInt))) == Z_NULL)
      {
        r = Z_MEM_ERROR;
        LEAVE
      }
      DUMPBITS(14)
      s->sub.trees.index = 0;
      Tracev((stderr, "inflate:       table sizes ok\n"));
      s->mode = BTREE;
    case BTREE:
      while (s->sub.trees.index < 4 + (s->sub.trees.table >> 10))
      {
        NEEDBITS(3)
        s->sub.trees.blens[border[s->sub.trees.index++]] = (uInt)b & 7;
        DUMPBITS(3)
      }
      while (s->sub.trees.index < 19)
        s->sub.trees.blens[border[s->sub.trees.index++]] = 0;
      s->sub.trees.bb = 7;
      t = inflate_trees_bits(s->sub.trees.blens, &s->sub.trees.bb,
                             &s->sub.trees.tb, s->hufts, z);
      if (t != Z_OK)
      {
        ZFREE(z, s->sub.trees.blens);
        r = t;
        if (r == Z_DATA_ERROR)
          s->mode = BAD;
        LEAVE
      }
      s->sub.trees.index = 0;
      Tracev((stderr, "inflate:       bits tree ok\n"));
      s->mode = DTREE;
    case DTREE:
      while (t = s->sub.trees.table,
             s->sub.trees.index < 258 + (t & 0x1f) + ((t >> 5) & 0x1f))
      {
        inflate_huft *h;
        uInt i, j, c;

        t = s->sub.trees.bb;
        NEEDBITS(t)
        h = s->sub.trees.tb + ((uInt)b & inflate_mask[t]);
        t = h->bits;
        c = h->base;
        if (c < 16)
        {
          DUMPBITS(t)
          s->sub.trees.blens[s->sub.trees.index++] = c;
        }
        else /* c == 16..18 */
        {
          i = c == 18 ? 7 : c - 14;
          j = c == 18 ? 11 : 3;
          NEEDBITS(t + i)
          DUMPBITS(t)
          j += (uInt)b & inflate_mask[i];
          DUMPBITS(i)
          i = s->sub.trees.index;
          t = s->sub.trees.table;
          if (i + j > 258 + (t & 0x1f) + ((t >> 5) & 0x1f) ||
              (c == 16 && i < 1))
          {
            ZFREE(z, s->sub.trees.blens);
            s->mode = BAD;
```

```
                 s->last ? " (last)" : ""));
          DUMPBITS(3)
          t = k & 7;                    /* go to byte boundary */
          DUMPBITS(t)
          s->mode = LENS;               /* get length of stored block */
          break;
        case 1:                         /* fixed */
          Tracev((stderr, "inflate:     fixed codes block%s\n",
                 s->last ? " (last)" : ""));
          {
            uInt bl, bd;
            inflate_huft *tl, *td;

            inflate_trees_fixed(&bl, &bd, &tl, &td, z);
            s->sub.decode.codes = inflate_codes_new(bl, bd, tl, td, z);
            if (s->sub.decode.codes == Z_NULL)
            {
              r = Z_MEM_ERROR;
              LEAVE
            }
          }
          DUMPBITS(3)
          s->mode = CODES;
          break;
        case 2:                         /* dynamic */
          Tracev((stderr, "inflate:     dynamic codes block%s\n",
                 s->last ? " (last)" : ""));
          DUMPBITS(3)
          s->mode = TABLE;
          break;
        case 3:                         /* illegal */
          DUMPBITS(3)
          s->mode = BAD;
          z->msg = (char*)"invalid block type";
          r = Z_DATA_ERROR;
          LEAVE
      }
      break;
    case LENS:
      NEEDBITS(32)
      if ((((~b) >> 16) & 0xffff) != (b & 0xffff))
      {
        s->mode = BAD;
        z->msg = (char*)"invalid stored block lengths";
        r = Z_DATA_ERROR;
        LEAVE
      }
      s->sub.left = (uInt)b & 0xffff;
      b = k = 0;                      /* dump bits */
      Tracev((stderr, "inflate:       stored length %u\n", s->sub.left));
      s->mode = s->sub.left ? STORED : (s->last ? DRY : TYPE);
      break;
    case STORED:
      if (n == 0)
        LEAVE
      NEEDOUT
      t = s->sub.left;
      if (t > n) t = n;
      if (t > m) t = m;
      zmemcpy(q, p, t);
      p += t;  n -= t;
      q += t;  m -= t;
      if ((s->sub.left -= t) != 0)
        break;
      Tracev((stderr, "inflate:       stored end, %lu total out\n",
              z->total_out + (q >= s->read ? q - s->read :
              (s->end - s->read) + (q - s->window))));
      s->mode = s->last ? DRY : TYPE;
      break;
    case TABLE:
      NEEDBITS(14)
      s->sub.trees.table = t = (uInt)b & 0x3fff;
#ifndef PKZIP_BUG_WORKAROUND
```

```
    *c = s->check;
  if (s->mode == BTREE || s->mode == DTREE)
    ZFREE(z, s->sub.trees.blens);
  if (s->mode == CODES)
    inflate_codes_free(s->sub.decode.codes, z);
  s->mode = TYPE;
  s->bitk = 0;
  s->bitb = 0;
  s->read = s->write = s->window;
  if (s->checkfn != Z_NULL)
    z->adler = s->check = (*s->checkfn)(0L, (const Bytef *)Z_NULL, 0);
  Tracev((stderr, "inflate:   blocks reset\n"));
}


inflate_blocks_statef *inflate_blocks_new(z, c, w)
z_streamp z;
check_func c;
uInt w;
{
  inflate_blocks_statef *s;

  if ((s = (inflate_blocks_statef *)ZALLOC
       (z,1,sizeof(struct inflate_blocks_state))) == Z_NULL)
    return s;
  if ((s->hufts =
       (inflate_huft *)ZALLOC(z, sizeof(inflate_huft), MANY)) == Z_NULL)
  {
    ZFREE(z, s);
    return Z_NULL;
  }
  if ((s->window = (Bytef *)ZALLOC(z, 1, w)) == Z_NULL)
  {
    ZFREE(z, s->hufts);
    ZFREE(z, s);
    return Z_NULL;
  }
  s->end = s->window + w;
  s->checkfn = c;
  s->mode = TYPE;
  Tracev((stderr, "inflate:   blocks allocated\n"));
  inflate_blocks_reset(s, z, Z_NULL);
  return s;
}


int inflate_blocks(s, z, r)
inflate_blocks_statef *s;
z_streamp z;
int r;
{
  uInt t;               /* temporary storage */
  uLong b;              /* bit buffer */
  uInt k;               /* bits in bit buffer */
  Bytef *p;             /* input data pointer */
  uInt n;               /* bytes available there */
  Bytef *q;             /* output window write pointer */
  uInt m;               /* bytes to end of window or read pointer */

  /* copy input/output information to locals (UPDATE macro restores) */
  LOAD

  /* process input based on current state */
  while (1) switch (s->mode)
  {
    case TYPE:
      NEEDBITS(3)
      t = (uInt)b & 7;
      s->last = t & 1;
      switch (t >> 1)
      {
        case 0:                         /* stored */
          Tracev((stderr, "inflate:     stored block%s\n",
/* infblock.c -- interpret and process block types to last block
 * Copyright (C) 1995-1998 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

#include "zutil.h"
#include "infblock.h"
#include "inftrees.h"
#include "infcodes.h"
#include "infutil.h"

struct inflate_codes_state {int dummy;}; /* for buggy compilers */

/* simplify the use of the inflate_huft type with some defines */
#define exop word.what.Exop
#define bits word.what.Bits

/* Table for deflate from PKZIP's appnote.txt. */
local const uInt border[] = { /* Order of the bit length code lengths */
        16, 17, 18, 0, 8, 7, 9, 6, 10, 5, 11, 4, 12, 3, 13, 2, 14, 1, 15};

/*
   Notes beyond the 1.93a appnote.txt:

   1. Distance pointers never point before the beginning of the output
      stream.
   2. Distance pointers can point back across blocks, up to 32k away.
   3. There is an implied maximum of 7 bits for the bit length table and
      15 bits for the actual data.
   4. If only one code exists, then it is encoded using one bit.  (Zero
      would be more efficient, but perhaps a little confusing.)  If two
      codes exist, they are coded using one bit each (0 and 1).
   5. There is no way of sending zero distance codes--a dummy must be
      sent if there are none.  (History: a pre 2.0 version of PKZIP would
      store blocks with no distance codes, but this was discovered to be
      too harsh a criterion.)  Valid only for 1.93a.  2.04c does allow
      zero distance codes, which is sent as one code of zero bits in
      length.
   6. There are up to 286 literal/length codes.  Code 256 represents the
      end-of-block.  Note however that the static length tree defines
      288 codes just to fill out the Huffman codes.  Codes 286 and 287
      cannot be used though, since there is no length base or extra bits
      defined for them.  Similarily, there are up to 30 distance codes.
      However, static trees define 32 codes (all 5 bits) to fill out the
      Huffman codes, but the last two had better not show up in the data.
   7. Unzip can check dynamic Huffman blocks for complete code sets.
      The exception is that a single code would not be complete (see #4).
   8. The five bits following the block type is really the number of
      literal codes sent minus 257.
   9. Length codes 8,16,16 are interpreted as 13 length codes of 8 bits
      (1+6+6).  Therefore, to output three times the length, you output
      three codes (1+1+1), whereas to output four times the same length,
      you only need two codes (1+3).  Hmm.
  10. In the tree reconstruction algorithm, Code = Code + Increment
      only if BitLength(i) is not zero.  (Pretty obvious.)
  11. Correction: 4 Bits: # of Bit Length codes - 4     (4 - 19)
  12. Note: length code 284 can represent 227-258, but length code 285
      really is 258.  The last length deserves its own, short code
      since it gets used a lot in very redundant files.  The length
      258 is special since 258 - 3 (the min match length) is 255.
  13. The literal/length and distance code bit lengths are read as a
      single stream of lengths.  It is possible (and advantageous) for
      a repeat code (16, 17, or 18) to go across the boundary between
      the two sets of lengths.
 */


void inflate_blocks_reset(s, z, c)
inflate_blocks_statef *s;
z_streamp z;
uLongf *c;
{
  if (c != Z_NULL)
```

```
/* infblock.h -- header to use infblock.c
 * Copyright (C) 1995-1998 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* WARNING: this file should *not* be used by applications. It is
   part of the implementation of the compression library and is
   subject to change. Applications should only use zlib.h.
 */

struct inflate_blocks_state;
typedef struct inflate_blocks_state FAR inflate_blocks_statef;

extern inflate_blocks_statef * inflate_blocks_new OF((
    z_streamp z,
    check_func c,              /* check function */
    uInt w));                  /* window size */

extern int inflate_blocks OF((
    inflate_blocks_statef *,
    z_streamp ,
    int));                     /* initial return code */

extern void inflate_blocks_reset OF((
    inflate_blocks_statef *,
    z_streamp ,
    uLongf *));                 /* check value on output */

extern int inflate_blocks_free OF((
    inflate_blocks_statef *,
    z_streamp));

extern void inflate_set_dictionary OF((
    inflate_blocks_statef *s,
    const Bytef *d,  /* dictionary */
    uInt  n));       /* dictionary length */

extern int inflate_blocks_sync_point OF((
    inflate_blocks_statef *s));
```

```
      OUTBYTE(c->sub.lit)
      c->mode = START;
      break;
    case WASH:           /* o: got eob, possibly more output */
      if (k > 7)         /* return unused byte, if any */
      {
        Assert(k < 16, "inflate_codes grabbed too many bytes")
        k -= 8;
        n++;
        p--;             /* can always return one */
      }
      FLUSH
      if (s->read != s->write)
        LEAVE
      c->mode = END;
    case END:
      r = Z_STREAM_END;
      LEAVE
    case BADCODE:        /* x: got error */
      r = Z_DATA_ERROR;
      LEAVE
    default:
      r = Z_STREAM_ERROR;
      LEAVE
  }
#ifdef NEED_DUMMY_RETURN
  return Z_STREAM_ERROR;  /* Some dumb compilers complain without this */
#endif
}


void inflate_codes_free(c, z)
inflate_codes_statef *c;
z_streamp z;
{
  ZFREE(z, c);
  Tracev((stderr, "inflate:       codes free\n"));
}
```

```
        break;
      }
      if (e & 32)               /* end of block */
      {
        Tracevv((stderr, "inflate:         end of block\n"));
        c->mode = WASH;
        break;
      }
      c->mode = BADCODE;        /* invalid code */
      z->msg = (char*)"invalid literal/length code";
      r = Z_DATA_ERROR;
      LEAVE
    case LENEXT:        /* i: getting length extra (have base) */
      j = c->sub.copy.get;
      NEEDBITS(j)
      c->len += (uInt)b & inflate_mask[j];
      DUMPBITS(j)
      c->sub.code.need = c->dbits;
      c->sub.code.tree = c->dtree;
      Tracevv((stderr, "inflate:         length %u\n", c->len));
      c->mode = DIST;
    case DIST:          /* i: get distance next */
      j = c->sub.code.need;
      NEEDBITS(j)
      t = c->sub.code.tree + ((uInt)b & inflate_mask[j]);
      DUMPBITS(t->bits)
      e = (uInt)(t->exop);
      if (e & 16)               /* distance */
      {
        c->sub.copy.get = e & 15;
        c->sub.copy.dist = t->base;
        c->mode = DISTEXT;
        break;
      }
      if ((e & 64) == 0)        /* next table */
      {
        c->sub.code.need = e;
        c->sub.code.tree = t + t->base;
        break;
      }
      c->mode = BADCODE;        /* invalid code */
      z->msg = (char*)"invalid distance code";
      r = Z_DATA_ERROR;
      LEAVE
    case DISTEXT:       /* i: getting distance extra */
      j = c->sub.copy.get;
      NEEDBITS(j)
      c->sub.copy.dist += (uInt)b & inflate_mask[j];
      DUMPBITS(j)
      Tracevv((stderr, "inflate:         distance %u\n", c->sub.copy.dist));
      c->mode = COPY;
    case COPY:          /* o: copying bytes in window, waiting for space */
#ifndef __TURBOC__ /* Turbo C bug for following expression */
      f = (uInt)(q - s->window) < c->sub.copy.dist ?
          s->end - (c->sub.copy.dist - (q - s->window)) :
          q - c->sub.copy.dist;
#else
      f = q - c->sub.copy.dist;
      if ((uInt)(q - s->window) < c->sub.copy.dist)
        f = s->end - (c->sub.copy.dist - (uInt)(q - s->window));
#endif
      while (c->len)
      {
        NEEDOUT
        OUTBYTE(*f++)
        if (f == s->end)
          f = s->window;
        c->len--;
      }
      c->mode = START;
      break;
    case LIT:           /* o: got literal, waiting for output space */
      NEEDOUT
    c->dtree = td;
    Tracev((stderr, "inflate:       codes new\n"));
  }
  return c;
}


int inflate_codes(s, z, r)
inflate_blocks_statef *s;
z_streamp z;
int r;
{
  uInt j;               /* temporary storage */
  inflate_huft *t;      /* temporary pointer */
  uInt e;               /* extra bits or operation */
  uLong b;              /* bit buffer */
  uInt k;               /* bits in bit buffer */
  Bytef *p;             /* input data pointer */
  uInt n;               /* bytes available there */
  Bytef *q;             /* output window write pointer */
  uInt m;               /* bytes to end of window or read pointer */
  Bytef *f;             /* pointer to copy strings from */
  inflate_codes_statef *c = s->sub.decode.codes;  /* codes state */

  /* copy input/output information to locals (UPDATE macro restores) */
  LOAD

  /* process input and output based on current state */
  while (1) switch (c->mode)
  {             /* waiting for "i:"=input, "o:"=output, "x:"=nothing */
    case START:         /* x: set up for LEN */
#ifndef SLOW
      if (m >= 258 && n >= 10)
      {
        UPDATE
        r = inflate_fast(c->lbits, c->dbits, c->ltree, c->dtree, s, z);
        LOAD
        if (r != Z_OK)
        {
          c->mode = r == Z_STREAM_END ? WASH : BADCODE;
          break;
        }
      }
#endif /* !SLOW */
      c->sub.code.need = c->lbits;
      c->sub.code.tree = c->ltree;
      c->mode = LEN;
    case LEN:           /* i: get length/literal/eob next */
      j = c->sub.code.need;
      NEEDBITS(j)
      t = c->sub.code.tree + ((uInt)b & inflate_mask[j]);
      DUMPBITS(t->bits)
      e = (uInt)(t->exop);
      if (e == 0)               /* literal */
      {
        c->sub.lit = t->base;
        Tracevv((stderr, t->base >= 0x20 && t->base < 0x7f ?
                 "inflate:         literal '%c'\n" :
                 "inflate:         literal 0x%02x\n", t->base));
        c->mode = LIT;
        break;
      }
      if (e & 16)               /* length */
      {
        c->sub.copy.get = e & 15;
        c->len = t->base;
        c->mode = LENEXT;
        break;
      }
      if ((e & 64) == 0)        /* next table */
      {
        c->sub.code.need = e;
        c->sub.code.tree = t + t->base;
```

```
/* infcodes.c -- process literals and length/distance pairs
 * Copyright (C) 1995-1998 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

#include "zutil.h"
#include "inftrees.h"
#include "infblock.h"
#include "infcodes.h"
#include "infutil.h"
#include "inffast.h"

/* simplify the use of the inflate_huft type with some defines */
#define exop word.what.Exop
#define bits word.what.Bits

typedef enum {        /* waiting for "i:"=input, "o:"=output, "x:"=nothing */
      START,    /* x: set up for LEN */
      LEN,      /* i: get length/literal/eob next */
      LENEXT,   /* i: getting length extra (have base) */
      DIST,     /* i: get distance next */
      DISTEXT,  /* i: getting distance extra */
      COPY,     /* o: copying bytes in window, waiting for space */
      LIT,      /* o: got literal, waiting for output space */
      WASH,     /* o: got eob, possibly still output waiting */
      END,      /* x: got eob and all data flushed */
      BADCODE}  /* x: got error */
inflate_codes_mode;

/* inflate codes private state */
struct inflate_codes_state {

  /* mode */
  inflate_codes_mode mode;      /* current inflate_codes mode */

  /* mode dependent information */
  uInt len;
  union {
    struct {
      inflate_huft *tree;       /* pointer into tree */
      uInt need;                /* bits needed */
    } code;             /* if LEN or DIST, where in tree */
    uInt lit;           /* if LIT, literal */
    struct {
      uInt get;                 /* bits to get for extra */
      uInt dist;                /* distance back to copy from */
    } copy;             /* if EXT or COPY, where and how much */
  } sub;                /* submode */

  /* mode independent information */
  Byte lbits;           /* ltree bits decoded per branch */
  Byte dbits;           /* dtree bits decoder per branch */
  inflate_huft *ltree;          /* literal/length/eob tree */
  inflate_huft *dtree;          /* distance tree */

};


inflate_codes_statef *inflate_codes_new(bl, bd, tl, td, z)
uInt bl, bd;
inflate_huft *tl;
inflate_huft *td; /* need separate declaration for Borland C++ */
z_streamp z;
{
  inflate_codes_statef *c;

  if ((c = (inflate_codes_statef *)
       ZALLOC(z,1,sizeof(struct inflate_codes_state))) != Z_NULL)
  {
    c->mode = START;
    c->lbits = (Byte)bl;
    c->dbits = (Byte)bd;
    c->ltree = tl;
```

```
        }
      }
      else if (e & 32)
      {
        Tracevv((stderr, "inflate:         * end of block\n"));
        UNGRAB
        UPDATE
        return Z_STREAM_END;
      }
      else
      {
        z->msg = (char*)"invalid literal/length code";
        UNGRAB
        UPDATE
        return Z_DATA_ERROR;
      }
    } while (1);
  } while (m >= 258 && n >= 10);

  /* not enough input or output--restore pointers and return */
  UNGRAB
  UPDATE
  return Z_OK;
}
```

```
        /* get extra bits for length */
        e &= 15;
        c = t->base + ((uInt)b & inflate_mask[e]);
        DUMPBITS(e)
        Tracevv((stderr, "inflate:         * length %u\n", c));

        /* decode distance base of block to copy */
        GRABBITS(15);           /* max bits for distance code */
        e = (t = td + ((uInt)b & md))->exop;
        do {
          DUMPBITS(t->bits)
          if (e & 16)
          {
            /* get extra bits to add to distance base */
            e &= 15;
            GRABBITS(e)         /* get extra bits (up to 13) */
            d = t->base + ((uInt)b & inflate_mask[e]);
            DUMPBITS(e)
            Tracevv((stderr, "inflate:         * distance %u\n", d));

            /* do the copy */
            m -= c;
            if ((uInt)(q - s->window) >= d)     /* offset before dest */
            {                                   /*  just copy */
              r = q - d;
              *q++ = *r++;  c--;        /* minimum count is three, */
              *q++ = *r++;  c--;        /*  so unroll loop a little */
            }
            else                        /* else offset after destination */
            {
              e = d - (uInt)(q - s->window); /* bytes from offset to end */
              r = s->end - e;           /* pointer to offset */
              if (c > e)                /* if source crosses, */
              {
                c -= e;                 /* copy to end of window */
                do {
                  *q++ = *r++;
                } while (--e);
                r = s->window;          /* copy rest from start of window */
              }
            }
            do {                        /* copy all or what's left */
              *q++ = *r++;
            } while (--c);
            break;
          }
          else if ((e & 64) == 0)
          {
            t += t->base;
            e = (t += ((uInt)b & inflate_mask[e]))->exop;
          }
          else
          {
            z->msg = (char*)"invalid distance code";
            UNGRAB
            UPDATE
            return Z_DATA_ERROR;
          }
        } while (1);
        break;
      }
      if ((e & 64) == 0)
      {
        t += t->base;
        if ((e = (t += ((uInt)b & inflate_mask[e]))->exop) == 0)
        {
          DUMPBITS(t->bits)
          Tracevv((stderr, t->base >= 0x20 && t->base < 0x7f ?
                    "inflate:         * literal '%c'\n" :
                    "inflate:         * literal 0x%02x\n", t->base));
          *q++ = (Byte)t->base;
          m--;
          break;
```

```
/* inffast.c -- process literals and length/distance pairs fast
 * Copyright (C) 1995-1998 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

#include "zutil.h"
#include "inftrees.h"
#include "infblock.h"
#include "infcodes.h"
#include "infutil.h"
#include "inffast.h"

struct inflate_codes_state {int dummy;}; /* for buggy compilers */

/* simplify the use of the inflate_huft type with some defines */
#define exop word.what.Exop
#define bits word.what.Bits

/* macros for bit input with no checking and for returning unused bytes */
#define GRABBITS(j) {while(k<(j)){b|=((uLong)NEXTBYTE)<<k;k+=8;}}
#define UNGRAB {c=z->avail_in-n;c=(k>>3)<c?k>>3:c;n+=c;p-=c;k-=c<<3;}

/* Called with number of bytes left to write in window at least 258
   (the maximum string length) and number of input bytes available
   at least ten.  The ten bytes are six bytes for the longest length/
   distance pair plus four bytes for overloading the bit buffer. */

int inflate_fast(bl, bd, tl, td, s, z)
uInt bl, bd;
inflate_huft *tl;
inflate_huft *td; /* need separate declaration for Borland C++ */
inflate_blocks_statef *s;
z_streamp z;
{
  inflate_huft *t;      /* temporary pointer */
  uInt e;               /* extra bits or operation */
  uLong b;              /* bit buffer */
  uInt k;               /* bits in bit buffer */
  Bytef *p;             /* input data pointer */
  uInt n;               /* bytes available there */
  Bytef *q;             /* output window write pointer */
  uInt m;               /* bytes to end of window or read pointer */
  uInt ml;              /* mask for literal/length tree */
  uInt md;              /* mask for distance tree */
  uInt c;               /* bytes to copy */
  uInt d;               /* distance back to copy from */
  Bytef *r;             /* copy source pointer */

  /* load input, output, bit values */
  LOAD

  /* initialize masks */
  ml = inflate_mask[bl];
  md = inflate_mask[bd];

  /* do until not enough input or output space for fast loop */
  do {                          /* assume called with m >= 258 && n >= 10 */
    /* get literal/length code */
    GRABBITS(20)                /* max bits for literal/length code */
    if ((e = (t = tl + ((uInt)b & ml))->exop) == 0)
    {
      DUMPBITS(t->bits)
      Tracevv((stderr, t->base >= 0x20 && t->base < 0x7f ?
                "inflate:         * literal '%c'\n" :
                "inflate:         * literal 0x%02x\n", t->base));
      *q++ = (Byte)t->base;
      m--;
      continue;
    }
    do {
      DUMPBITS(t->bits)
      if (e & 16)
      {
```

```
/* inffast.h -- header to use inffast.c
 * Copyright (C) 1995-1998 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* WARNING: this file should *not* be used by applications. It is
   part of the implementation of the compression library and is
   subject to change. Applications should only use zlib.h.
 */

extern int inflate_fast OF((
    uInt,
    uInt,
    inflate_huft *,
    inflate_huft *,
    inflate_blocks_statef *,
    z_streamp ));
    {{{80,5}},2}, {{{87,5}},385}, {{{83,5}},25}, {{{91,5}},6145},
    {{{81,5}},7}, {{{89,5}},1537}, {{{85,5}},97}, {{{93,5}},24577},
    {{{80,5}},4}, {{{88,5}},769}, {{{84,5}},49}, {{{92,5}},12289},
    {{{82,5}},13}, {{{90,5}},3073}, {{{86,5}},193}, {{{192,5}},24577}
  };
```

```
    {{{84,7}},99}, {{{0,8}},127}, {{{0,8}},63}, {{{0,9}},222},
    {{{82,7}},27}, {{{0,8}},111}, {{{0,8}},47}, {{{0,9}},190},
    {{{0,8}},15}, {{{0,8}},143}, {{{0,8}},79}, {{{0,9}},254},
    {{{96,7}},256}, {{{0,8}},80}, {{{0,8}},16}, {{{84,8}},115},
    {{{82,7}},31}, {{{0,8}},112}, {{{0,8}},48}, {{{0,9}},193},
    {{{80,7}},10}, {{{0,8}},96}, {{{0,8}},32}, {{{0,9}},161},
    {{{0,8}},0}, {{{0,8}},128}, {{{0,8}},64}, {{{0,9}},225},
    {{{80,7}},6}, {{{0,8}},88}, {{{0,8}},24}, {{{0,9}},145},
    {{{83,7}},59}, {{{0,8}},120}, {{{0,8}},56}, {{{0,9}},209},
    {{{81,7}},17}, {{{0,8}},104}, {{{0,8}},40}, {{{0,9}},177},
    {{{0,8}},8}, {{{0,8}},136}, {{{0,8}},72}, {{{0,9}},241},
    {{{80,7}},4}, {{{0,8}},84}, {{{0,8}},20}, {{{85,8}},227},
    {{{83,7}},43}, {{{0,8}},116}, {{{0,8}},52}, {{{0,9}},201},
    {{{81,7}},13}, {{{0,8}},100}, {{{0,8}},36}, {{{0,9}},169},
    {{{0,8}},4}, {{{0,8}},132}, {{{0,8}},68}, {{{0,9}},233},
    {{{80,7}},8}, {{{0,8}},92}, {{{0,8}},28}, {{{0,9}},153},
    {{{84,7}},83}, {{{0,8}},124}, {{{0,8}},60}, {{{0,9}},217},
    {{{82,7}},23}, {{{0,8}},108}, {{{0,8}},44}, {{{0,9}},185},
    {{{0,8}},12}, {{{0,8}},140}, {{{0,8}},76}, {{{0,9}},249},
    {{{80,7}},3}, {{{0,8}},82}, {{{0,8}},18}, {{{85,8}},163},
    {{{83,7}},35}, {{{0,8}},114}, {{{0,8}},50}, {{{0,9}},197},
    {{{81,7}},11}, {{{0,8}},98}, {{{0,8}},34}, {{{0,9}},165},
    {{{0,8}},2}, {{{0,8}},130}, {{{0,8}},66}, {{{0,9}},229},
    {{{80,7}},7}, {{{0,8}},90}, {{{0,8}},26}, {{{0,9}},149},
    {{{84,7}},67}, {{{0,8}},122}, {{{0,8}},58}, {{{0,9}},213},
    {{{82,7}},19}, {{{0,8}},106}, {{{0,8}},42}, {{{0,9}},181},
    {{{0,8}},10}, {{{0,8}},138}, {{{0,8}},74}, {{{0,9}},245},
    {{{80,7}},5}, {{{0,8}},86}, {{{0,8}},22}, {{{192,8}},0},
    {{{83,7}},51}, {{{0,8}},118}, {{{0,8}},54}, {{{0,9}},205},
    {{{81,7}},15}, {{{0,8}},102}, {{{0,8}},38}, {{{0,9}},173},
    {{{0,8}},6}, {{{0,8}},134}, {{{0,8}},70}, {{{0,9}},237},
    {{{80,7}},9}, {{{0,8}},94}, {{{0,8}},30}, {{{0,9}},157},
    {{{84,7}},99}, {{{0,8}},126}, {{{0,8}},62}, {{{0,9}},221},
    {{{82,7}},27}, {{{0,8}},110}, {{{0,8}},46}, {{{0,9}},189},
    {{{0,8}},14}, {{{0,8}},142}, {{{0,8}},78}, {{{0,9}},253},
    {{{96,7}},256}, {{{0,8}},81}, {{{0,8}},17}, {{{85,8}},131},
    {{{82,7}},31}, {{{0,8}},113}, {{{0,8}},49}, {{{0,9}},195},
    {{{80,7}},10}, {{{0,8}},97}, {{{0,8}},33}, {{{0,9}},163},
    {{{0,8}},1}, {{{0,8}},129}, {{{0,8}},65}, {{{0,9}},227},
    {{{80,7}},6}, {{{0,8}},89}, {{{0,8}},25}, {{{0,9}},147},
    {{{83,7}},59}, {{{0,8}},121}, {{{0,8}},57}, {{{0,9}},211},
    {{{81,7}},17}, {{{0,8}},105}, {{{0,8}},41}, {{{0,9}},179},
    {{{0,8}},9}, {{{0,8}},137}, {{{0,8}},73}, {{{0,9}},243},
    {{{80,7}},4}, {{{0,8}},85}, {{{0,8}},21}, {{{80,8}},258},
    {{{83,7}},43}, {{{0,8}},117}, {{{0,8}},53}, {{{0,9}},203},
    {{{81,7}},13}, {{{0,8}},101}, {{{0,8}},37}, {{{0,9}},171},
    {{{0,8}},5}, {{{0,8}},133}, {{{0,8}},69}, {{{0,9}},235},
    {{{80,7}},8}, {{{0,8}},93}, {{{0,8}},29}, {{{0,9}},155},
    {{{84,7}},83}, {{{0,8}},125}, {{{0,8}},61}, {{{0,9}},219},
    {{{82,7}},23}, {{{0,8}},109}, {{{0,8}},45}, {{{0,9}},187},
    {{{0,8}},13}, {{{0,8}},141}, {{{0,8}},77}, {{{0,9}},251},
    {{{80,7}},3}, {{{0,8}},83}, {{{0,8}},19}, {{{85,8}},195},
    {{{83,7}},35}, {{{0,8}},115}, {{{0,8}},51}, {{{0,9}},199},
    {{{81,7}},11}, {{{0,8}},99}, {{{0,8}},35}, {{{0,9}},167},
    {{{0,8}},3}, {{{0,8}},131}, {{{0,8}},67}, {{{0,9}},231},
    {{{80,7}},7}, {{{0,8}},91}, {{{0,8}},27}, {{{0,9}},151},
    {{{84,7}},67}, {{{0,8}},123}, {{{0,8}},59}, {{{0,9}},215},
    {{{82,7}},19}, {{{0,8}},107}, {{{0,8}},43}, {{{0,9}},183},
    {{{0,8}},11}, {{{0,8}},139}, {{{0,8}},75}, {{{0,9}},247},
    {{{80,7}},5}, {{{0,8}},87}, {{{0,8}},23}, {{{192,8}},0},
    {{{83,7}},51}, {{{0,8}},119}, {{{0,8}},55}, {{{0,9}},207},
    {{{81,7}},15}, {{{0,8}},103}, {{{0,8}},39}, {{{0,9}},175},
    {{{0,8}},7}, {{{0,8}},135}, {{{0,8}},71}, {{{0,9}},239},
    {{{80,7}},9}, {{{0,8}},95}, {{{0,8}},31}, {{{0,9}},159},
    {{{84,7}},99}, {{{0,8}},127}, {{{0,8}},63}, {{{0,9}},223},
    {{{82,7}},27}, {{{0,8}},111}, {{{0,8}},47}, {{{0,9}},191},
    {{{0,8}},15}, {{{0,8}},143}, {{{0,8}},79}, {{{0,9}},255}
  };
local inflate_huft fixed_td[] = {
    {{{80,5}},1}, {{{87,5}},257}, {{{83,5}},17}, {{{91,5}},4097},
    {{{81,5}},5}, {{{89,5}},1025}, {{{85,5}},65}, {{{93,5}},16385},
    {{{80,5}},3}, {{{88,5}},513}, {{{84,5}},33}, {{{92,5}},8193},
    {{{82,5}},9}, {{{90,5}},2049}, {{{86,5}},129}, {{{192,5}},24577},
/* inffixed.h -- table for decoding fixed codes
 * Generated automatically by the maketree.c program
 */

/* WARNING: this file should *not* be used by applications. It is
   part of the implementation of the compression library and is
   subject to change. Applications should only use zlib.h.
 */

local uInt fixed_bl = 9;
local uInt fixed_bd = 5;
local inflate_huft fixed_tl[] = {
    {{{96,7}},256}, {{{0,8}},80}, {{{0,8}},16}, {{{84,8}},115},
    {{{82,7}},31}, {{{0,8}},112}, {{{0,8}},48}, {{{0,9}},192},
    {{{80,7}},10}, {{{0,8}},96}, {{{0,8}},32}, {{{0,9}},160},
    {{{0,8}},0}, {{{0,8}},128}, {{{0,8}},64}, {{{0,9}},224},
    {{{80,7}},6}, {{{0,8}},88}, {{{0,8}},24}, {{{0,9}},144},
    {{{83,7}},59}, {{{0,8}},120}, {{{0,8}},56}, {{{0,9}},208},
    {{{81,7}},17}, {{{0,8}},104}, {{{0,8}},40}, {{{0,9}},176},
    {{{0,8}},8}, {{{0,8}},136}, {{{0,8}},72}, {{{0,9}},240},
    {{{80,7}},4}, {{{0,8}},84}, {{{0,8}},20}, {{{85,8}},227},
    {{{83,7}},43}, {{{0,8}},116}, {{{0,8}},52}, {{{0,9}},200},
    {{{81,7}},13}, {{{0,8}},100}, {{{0,8}},36}, {{{0,9}},168},
    {{{0,8}},4}, {{{0,8}},132}, {{{0,8}},68}, {{{0,9}},232},
    {{{80,7}},8}, {{{0,8}},92}, {{{0,8}},28}, {{{0,9}},152},
    {{{84,7}},83}, {{{0,8}},124}, {{{0,8}},60}, {{{0,9}},216},
    {{{82,7}},23}, {{{0,8}},108}, {{{0,8}},44}, {{{0,9}},184},
    {{{0,8}},12}, {{{0,8}},140}, {{{0,8}},76}, {{{0,9}},248},
    {{{80,7}},3}, {{{0,8}},82}, {{{0,8}},18}, {{{85,8}},163},
    {{{83,7}},35}, {{{0,8}},114}, {{{0,8}},50}, {{{0,9}},196},
    {{{81,7}},11}, {{{0,8}},98}, {{{0,8}},34}, {{{0,9}},164},
    {{{0,8}},2}, {{{0,8}},130}, {{{0,8}},66}, {{{0,9}},228},
    {{{80,7}},7}, {{{0,8}},90}, {{{0,8}},26}, {{{0,9}},148},
    {{{84,7}},67}, {{{0,8}},122}, {{{0,8}},58}, {{{0,9}},212},
    {{{82,7}},19}, {{{0,8}},106}, {{{0,8}},42}, {{{0,9}},180},
    {{{0,8}},10}, {{{0,8}},138}, {{{0,8}},74}, {{{0,9}},244},
    {{{80,7}},5}, {{{0,8}},86}, {{{0,8}},22}, {{{192,8}},0},
    {{{83,7}},51}, {{{0,8}},118}, {{{0,8}},54}, {{{0,9}},204},
    {{{81,7}},15}, {{{0,8}},102}, {{{0,8}},38}, {{{0,9}},172},
    {{{0,8}},6}, {{{0,8}},134}, {{{0,8}},70}, {{{0,9}},236},
    {{{80,7}},9}, {{{0,8}},94}, {{{0,8}},30}, {{{0,9}},156},
    {{{84,7}},99}, {{{0,8}},126}, {{{0,8}},62}, {{{0,9}},220},
    {{{82,7}},27}, {{{0,8}},110}, {{{0,8}},46}, {{{0,9}},188},
    {{{0,8}},14}, {{{0,8}},142}, {{{0,8}},78}, {{{0,9}},252},
    {{{96,7}},256}, {{{0,8}},81}, {{{0,8}},17}, {{{85,8}},131},
    {{{82,7}},31}, {{{0,8}},113}, {{{0,8}},49}, {{{0,9}},194},
    {{{80,7}},10}, {{{0,8}},97}, {{{0,8}},33}, {{{0,9}},162},
    {{{0,8}},1}, {{{0,8}},129}, {{{0,8}},65}, {{{0,9}},226},
    {{{80,7}},6}, {{{0,8}},89}, {{{0,8}},25}, {{{0,9}},146},
    {{{83,7}},59}, {{{0,8}},121}, {{{0,8}},57}, {{{0,9}},210},
    {{{81,7}},17}, {{{0,8}},105}, {{{0,8}},41}, {{{0,9}},178},
    {{{0,8}},9}, {{{0,8}},137}, {{{0,8}},73}, {{{0,9}},242},
    {{{80,7}},4}, {{{0,8}},85}, {{{0,8}},21}, {{{80,8}},258},
    {{{83,7}},43}, {{{0,8}},117}, {{{0,8}},53}, {{{0,9}},202},
    {{{81,7}},13}, {{{0,8}},101}, {{{0,8}},37}, {{{0,9}},170},
    {{{0,8}},5}, {{{0,8}},133}, {{{0,8}},69}, {{{0,9}},234},
    {{{80,7}},8}, {{{0,8}},93}, {{{0,8}},29}, {{{0,9}},154},
    {{{84,7}},83}, {{{0,8}},125}, {{{0,8}},61}, {{{0,9}},218},
    {{{82,7}},23}, {{{0,8}},109}, {{{0,8}},45}, {{{0,9}},186},
    {{{0,8}},13}, {{{0,8}},141}, {{{0,8}},77}, {{{0,9}},250},
    {{{80,7}},3}, {{{0,8}},83}, {{{0,8}},19}, {{{85,8}},195},
    {{{83,7}},35}, {{{0,8}},115}, {{{0,8}},51}, {{{0,9}},198},
    {{{81,7}},11}, {{{0,8}},99}, {{{0,8}},35}, {{{0,9}},166},
    {{{0,8}},3}, {{{0,8}},131}, {{{0,8}},67}, {{{0,9}},230},
    {{{80,7}},7}, {{{0,8}},91}, {{{0,8}},27}, {{{0,9}},150},
    {{{84,7}},67}, {{{0,8}},123}, {{{0,8}},59}, {{{0,9}},214},
    {{{82,7}},19}, {{{0,8}},107}, {{{0,8}},43}, {{{0,9}},182},
    {{{0,8}},11}, {{{0,8}},139}, {{{0,8}},75}, {{{0,9}},246},
    {{{80,7}},5}, {{{0,8}},87}, {{{0,8}},23}, {{{192,8}},0},
    {{{83,7}},51}, {{{0,8}},119}, {{{0,8}},55}, {{{0,9}},206},
    {{{81,7}},15}, {{{0,8}},103}, {{{0,8}},39}, {{{0,9}},174},
    {{{0,8}},7}, {{{0,8}},135}, {{{0,8}},71}, {{{0,9}},238},
    {{{80,7}},9}, {{{0,8}},95}, {{{0,8}},31}, {{{0,9}},158},
}
```

```
    length = (1<<z->state->wbits)-1;
    dictionary += dictLength - length;
  }
  inflate_set_dictionary(z->state->blocks, dictionary, length);
  z->state->mode = BLOCKS;
  return Z_OK;
}


int ZEXPORT inflateSync(z)
z_streamp z;
{
  uInt n;       /* number of bytes to look at */
  Bytef *p;     /* pointer to bytes */
  uInt m;       /* number of marker bytes found in a row */
  uLong r, w;   /* temporaries to save total_in and total_out */

  /* set up */
  if (z == Z_NULL || z->state == Z_NULL)
    return Z_STREAM_ERROR;
  if (z->state->mode != BAD)
  {
    z->state->mode = BAD;
    z->state->sub.marker = 0;
  }
  if ((n = z->avail_in) == 0)
    return Z_BUF_ERROR;
  p = z->next_in;
  m = z->state->sub.marker;

  /* search */
  while (n && m < 4)
  {
    static const Byte mark[4] = {0, 0, 0xff, 0xff};
    if (*p == mark[m])
      m++;
    else if (*p)
      m = 0;
    else
      m = 4 - m;
    p++, n--;
  }

  /* restore */
  z->total_in += p - z->next_in;
  z->next_in = p;
  z->avail_in = n;
  z->state->sub.marker = m;

  /* return no joy or set up to restart on a new block */
  if (m != 4)
    return Z_DATA_ERROR;
  r = z->total_in;  w = z->total_out;
  inflateReset(z);
  z->total_in = r;  z->total_out = w;
  z->state->mode = BLOCKS;
  return Z_OK;
}


/* Returns true if inflate is currently at the end of a block generated
 * by Z_SYNC_FLUSH or Z_FULL_FLUSH. This function is used by one PPP
 * implementation to provide an additional safety check. PPP uses Z_SYNC_FLUSH
 * but removes the length bytes of the resulting empty stored block. When
 * decompressing, PPP checks that at the end of input packet, inflate is
 * waiting for these length bytes.
 */
int ZEXPORT inflateSyncPoint(z)
z_streamp z;
{
  if (z == Z_NULL || z->state == Z_NULL || z->state->blocks == Z_NULL)
    return Z_STREAM_ERROR;
  return inflate_blocks_sync_point(z->state->blocks);
```

```
    case BLOCKS:
      r = inflate_blocks(z->state->blocks, z, r);
      if (r == Z_DATA_ERROR)
      {
        z->state->mode = BAD;
        z->state->sub.marker = 0;       /* can try inflateSync */
        break;
      }
      if (r == Z_OK)
        r = f;
      if (r != Z_STREAM_END)
        return r;
      r = f;
      inflate_blocks_reset(z->state->blocks, z, &z->state->sub.check.was);
      if (z->state->nowrap)
      {
        z->state->mode = DONE;
        break;
      }
      z->state->mode = CHECK4;
    case CHECK4:
      NEEDBYTE
      z->state->sub.check.need = (uLong)NEXTBYTE << 24;
      z->state->mode = CHECK3;
    case CHECK3:
      NEEDBYTE
      z->state->sub.check.need += (uLong)NEXTBYTE << 16;
      z->state->mode = CHECK2;
    case CHECK2:
      NEEDBYTE
      z->state->sub.check.need += (uLong)NEXTBYTE << 8;
      z->state->mode = CHECK1;
    case CHECK1:
      NEEDBYTE
      z->state->sub.check.need += (uLong)NEXTBYTE;

      if (z->state->sub.check.was != z->state->sub.check.need)
      {
        z->state->mode = BAD;
        z->msg = (char*)"incorrect data check";
        z->state->sub.marker = 5;       /* can't try inflateSync */
        break;
      }
      Tracev((stderr, "inflate: zlib check ok\n"));
      z->state->mode = DONE;
    case DONE:
      return Z_STREAM_END;
    case BAD:
      return Z_DATA_ERROR;
    default:
      return Z_STREAM_ERROR;
  }
#ifdef NEED_DUMMY_RETURN
  return Z_STREAM_ERROR;  /* Some dumb compilers complain without this */
#endif
}


int ZEXPORT inflateSetDictionary(z, dictionary, dictLength)
z_streamp z;
const Bytef *dictionary;
uInt  dictLength;
{
  uInt length = dictLength;

  if (z == Z_NULL || z->state == Z_NULL || z->state->mode != DICT0)
    return Z_STREAM_ERROR;

  if (adler32(1L, dictionary, dictLength) != z->adler) return Z_DATA_ERROR;
  z->adler = 1L;

  if (length >= ((uInt)1<<z->state->wbits))
  {
```

```
#define NEEDBYTE {if(z->avail_in==0)return r;r=f;}
#define NEXTBYTE (z->avail_in--,z->total_in++,*z->next_in++)

int ZEXPORT inflate(z, f)
z_streamp z;
int f;
{
  int r;
  uInt b;

  if (z == Z_NULL || z->state == Z_NULL || z->next_in == Z_NULL)
    return Z_STREAM_ERROR;
  f = f == Z_FINISH ? Z_BUF_ERROR : Z_OK;
  r = Z_BUF_ERROR;
  while (1) switch (z->state->mode)
  {
    case METHOD:
      NEEDBYTE
      if (((z->state->sub.method = NEXTBYTE) & 0xf) != Z_DEFLATED)
      {
        z->state->mode = BAD;
        z->msg = (char*)"unknown compression method";
        z->state->sub.marker = 5;       /* can't try inflateSync */
        break;
      }
      if ((z->state->sub.method >> 4) + 8 > z->state->wbits)
      {
        z->state->mode = BAD;
        z->msg = (char*)"invalid window size";
        z->state->sub.marker = 5;       /* can't try inflateSync */
        break;
      }
      z->state->mode = FLAG;
    case FLAG:
      NEEDBYTE
      b = NEXTBYTE;
      if (((z->state->sub.method << 8) + b) % 31)
      {
        z->state->mode = BAD;
        z->msg = (char*)"incorrect header check";
        z->state->sub.marker = 5;       /* can't try inflateSync */
        break;
      }
      Tracev((stderr, "inflate: zlib header ok\n"));
      if (!(b & PRESET_DICT))
      {
        z->state->mode = BLOCKS;
        break;
      }
      z->state->mode = DICT4;
    case DICT4:
      NEEDBYTE
      z->state->sub.check.need = (uLong)NEXTBYTE << 24;
      z->state->mode = DICT3;
    case DICT3:
      NEEDBYTE
      z->state->sub.check.need += (uLong)NEXTBYTE << 16;
      z->state->mode = DICT2;
    case DICT2:
      NEEDBYTE
      z->state->sub.check.need += (uLong)NEXTBYTE << 8;
      z->state->mode = DICT1;
    case DICT1:
      NEEDBYTE
      z->state->sub.check.need += (uLong)NEXTBYTE;
      z->adler = z->state->sub.check.need;
      z->state->mode = DICT0;
      return Z_NEED_DICT;
    case DICT0:
      z->state->mode = BAD;
      z->msg = (char*)"need dictionary";
      z->state->sub.marker = 0;       /* can try inflateSync */
      return Z_STREAM_ERROR;
  ZFREE(z, z->state);
  z->state = Z_NULL;
  Tracev((stderr, "inflate: end\n"));
  return Z_OK;
}


int ZEXPORT inflateInit2_(z, w, version, stream_size)
z_streamp z;
int w;
const char *version;
int stream_size;
{
  if (version == Z_NULL || version[0] != ZLIB_VERSION[0] ||
      stream_size != sizeof(z_stream))
      return Z_VERSION_ERROR;

  /* initialize state */
  if (z == Z_NULL)
    return Z_STREAM_ERROR;
  z->msg = Z_NULL;
  if (z->zalloc == Z_NULL)
  {
    z->zalloc = zcalloc;
    z->opaque = (voidpf)0;
  }
  if (z->zfree == Z_NULL) z->zfree = zcfree;
  if ((z->state = (struct internal_state FAR *)
       ZALLOC(z,1,sizeof(struct internal_state))) == Z_NULL)
    return Z_MEM_ERROR;
  z->state->blocks = Z_NULL;

  /* handle undocumented nowrap option (no zlib header or check) */
  z->state->nowrap = 0;
  if (w < 0)
  {
    w = - w;
    z->state->nowrap = 1;
  }

  /* set window size */
  if (w < 8 || w > 15)
  {
    inflateEnd(z);
    return Z_STREAM_ERROR;
  }
  z->state->wbits = (uInt)w;

  /* create inflate_blocks state */
  if ((z->state->blocks =
      inflate_blocks_new(z, z->state->nowrap ? Z_NULL : adler32, (uInt)1 << w))
      == Z_NULL)
  {
    inflateEnd(z);
    return Z_MEM_ERROR;
  }
  Tracev((stderr, "inflate: allocated\n"));

  /* reset state */
  inflateReset(z);
  return Z_OK;
}


int ZEXPORT inflateInit_(z, version, stream_size)
z_streamp z;
const char *version;
int stream_size;
{
  return inflateInit2_(z, DEF_WBITS, version, stream_size);
}
```

```
/* inflate.c -- zlib interface to inflate modules
 * Copyright (C) 1995-1998 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

#include "zutil.h"
#include "infblock.h"

struct inflate_blocks_state {int dummy;}; /* for buggy compilers */

typedef enum {
      METHOD,   /* waiting for method byte */
      FLAG,     /* waiting for flag byte */
      DICT4,    /* four dictionary check bytes to go */
      DICT3,    /* three dictionary check bytes to go */
      DICT2,    /* two dictionary check bytes to go */
      DICT1,    /* one dictionary check byte to go */
      DICT0,    /* waiting for inflateSetDictionary */
      BLOCKS,   /* decompressing blocks */
      CHECK4,   /* four check bytes to go */
      CHECK3,   /* three check bytes to go */
      CHECK2,   /* two check bytes to go */
      CHECK1,   /* one check byte to go */
      DONE,     /* finished check, done */
      BAD}      /* got an error--stay here */
inflate_mode;

/* inflate private state */
struct internal_state {

  /* mode */
  inflate_mode  mode;   /* current inflate mode */

  /* mode dependent information */
  union {
    uInt method;        /* if FLAGS, method byte */
    struct {
      uLong was;                /* computed check value */
      uLong need;               /* stream check value */
    } check;            /* if CHECK, check values to compare */
    uInt marker;        /* if BAD, inflateSync's marker bytes count */
  } sub;        /* submode */

  /* mode independent information */
  int  nowrap;          /* flag for no wrapper */
  uInt wbits;           /* log2(window size)  (8..15, defaults to 15) */
  inflate_blocks_statef
    *blocks;            /* current inflate_blocks state */

};


int ZEXPORT inflateReset(z)
z_streamp z;
{
  if (z == Z_NULL || z->state == Z_NULL)
    return Z_STREAM_ERROR;
  z->total_in = z->total_out = 0;
  z->msg = Z_NULL;
  z->state->mode = z->state->nowrap ? BLOCKS : METHOD;
  inflate_blocks_reset(z->state->blocks, z, Z_NULL);
  Tracev((stderr, "inflate: reset\n"));
  return Z_OK;
}


int ZEXPORT inflateEnd(z)
z_streamp z;
{
  if (z == Z_NULL || z->state == Z_NULL || z->zfree == Z_NULL)
    return Z_STREAM_ERROR;
  if (z->state->blocks != Z_NULL)
    inflate_blocks_free(z->state->blocks, z);
```

```
      c[k] = 5;
    fixed_bd = 5;
    huft_build(c, 30, 0, cpdist, cpdext, &fixed_td, &fixed_bd,
               fixed_mem, &f, v);

    /* done */
    ZFREE(z, v);
    ZFREE(z, c);
    fixed_built = 1;
  }
#endif
  *bl = fixed_bl;
  *bd = fixed_bd;
  *tl = fixed_tl;
  *td = fixed_td;
  return Z_OK;
}
```

```
      z->msg = (char*)"incomplete distance tree";
      r = Z_DATA_ERROR;
    }
    else if (r != Z_MEM_ERROR)
    {
      z->msg = (char*)"empty distance tree with lengths";
      r = Z_DATA_ERROR;
    }
    ZFREE(z, v);
    return r;
#endif
  }

  /* done */
  ZFREE(z, v);
  return Z_OK;
}


/* build fixed tables only once--keep them here */
#ifdef BUILDFIXED
local int fixed_built = 0;
#define FIXEDH 544      /* number of hufts used by fixed tables */
local inflate_huft fixed_mem[FIXEDH];
local uInt fixed_bl;
local uInt fixed_bd;
local inflate_huft *fixed_tl;
local inflate_huft *fixed_td;
#else
#include "inffixed.h"
#endif


int inflate_trees_fixed(bl, bd, tl, td, z)
uIntf *bl;               /* literal desired/actual bit depth */
uIntf *bd;               /* distance desired/actual bit depth */
inflate_huft * FAR *tl;  /* literal/length tree result */
inflate_huft * FAR *td;  /* distance tree result */
z_streamp z;             /* for memory allocation */
{
#ifdef BUILDFIXED
  /* build fixed tables if not already */
  if (!fixed_built)
  {
    int k;              /* temporary variable */
    uInt f = 0;         /* number of hufts used in fixed_mem */
    uIntf *c;           /* length list for huft_build */
    uIntf *v;           /* work area for huft_build */

    /* allocate memory */
    if ((c = (uIntf*)ZALLOC(z, 288, sizeof(uInt))) == Z_NULL)
      return Z_MEM_ERROR;
    if ((v = (uIntf*)ZALLOC(z, 288, sizeof(uInt))) == Z_NULL)
    {
      ZFREE(z, c);
      return Z_MEM_ERROR;
    }

    /* literal table */
    for (k = 0; k < 144; k++)
      c[k] = 8;
    for (; k < 256; k++)
      c[k] = 9;
    for (; k < 280; k++)
      c[k] = 7;
    for (; k < 288; k++)
      c[k] = 8;
    fixed_bl = 9;
    huft_build(c, 288, 257, cplens, cplext, &fixed_tl, &fixed_bl,
               fixed_mem, &f, v);

    /* distance table */
    for (k = 0; k < 30; k++)
```

```
int inflate_trees_bits(c, bb, tb, hp, z)
uIntf *c;               /* 19 code lengths */
uIntf *bb;              /* bits tree desired/actual depth */
inflate_huft * FAR *tb; /* bits tree result */
inflate_huft *hp;       /* space for trees */
z_streamp z;            /* for messages */
{
  int r;
  uInt hn = 0;          /* hufts used in space */
  uIntf *v;             /* work area for huft_build */

  if ((v = (uIntf*)ZALLOC(z, 19, sizeof(uInt))) == Z_NULL)
    return Z_MEM_ERROR;
  r = huft_build(c, 19, 19, (uIntf*)Z_NULL, (uIntf*)Z_NULL,
                 tb, bb, hp, &hn, v);
  if (r == Z_DATA_ERROR)
    z->msg = (char*)"oversubscribed dynamic bit lengths tree";
  else if (r == Z_BUF_ERROR || *bb == 0)
  {
    z->msg = (char*)"incomplete dynamic bit lengths tree";
    r = Z_DATA_ERROR;
  }
  ZFREE(z, v);
  return r;
}


int inflate_trees_dynamic(nl, nd, c, bl, bd, tl, td, hp, z)
uInt nl;                /* number of literal/length codes */
uInt nd;                /* number of distance codes */
uIntf *c;               /* that many (total) code lengths */
uIntf *bl;              /* literal desired/actual bit depth */
uIntf *bd;              /* distance desired/actual bit depth */
inflate_huft * FAR *tl; /* literal/length tree result */
inflate_huft * FAR *td; /* distance tree result */
inflate_huft *hp;       /* space for trees */
z_streamp z;            /* for messages */
{
  int r;
  uInt hn = 0;          /* hufts used in space */
  uIntf *v;             /* work area for huft_build */

  /* allocate work area */
  if ((v = (uIntf*)ZALLOC(z, 288, sizeof(uInt))) == Z_NULL)
    return Z_MEM_ERROR;

  /* build literal/length tree */
  r = huft_build(c, nl, 257, cplens, cplext, tl, bl, hp, &hn, v);
  if (r != Z_OK || *bl == 0)
  {
    if (r == Z_DATA_ERROR)
      z->msg = (char*)"oversubscribed literal/length tree";
    else if (r != Z_MEM_ERROR)
    {
      z->msg = (char*)"incomplete literal/length tree";
      r = Z_DATA_ERROR;
    }
    ZFREE(z, v);
    return r;
  }

  /* build distance tree */
  r = huft_build(c + nl, nd, 0, cpdist, cpdext, td, bd, hp, &hn, v);
  if (r != Z_OK || (*bd == 0 && nl > 257))
  {
    if (r == Z_DATA_ERROR)
      z->msg = (char*)"oversubscribed distance tree";
    else if (r == Z_BUF_ERROR) {
#ifdef PKZIP_BUG_WORKAROUND
      r = Z_OK;
    }
#else
```

```
          f -= a + 1;           /* deduct codes from patterns left */
          xp = c + k;
          if (j < z)
            while (++j < z)     /* try smaller tables up to z bits */
            {
              if ((f <<= 1) <= *++xp)
                break;          /* enough codes to use up j bits */
              f -= *xp;         /* else deduct codes from patterns */
            }
        }
        z = 1 << j;             /* table entries for j-bit table */

        /* allocate new table */
        if (*hn + z > MANY)     /* (note: doesn't matter for fixed) */
          return Z_MEM_ERROR;   /* not enough memory */
        u[h] = q = hp + *hn;
        *hn += z;

        /* connect to last table, if there is one */
        if (h)
        {
          x[h] = i;             /* save pattern for backing up */
          r.bits = (Byte)l;     /* bits to dump before this table */
          r.exop = (Byte)j;     /* bits in this table */
          j = i >> (w - l);
          r.base = (uInt)(q - u[h-1] - j);   /* offset to this table */
          u[h-1][j] = r;        /* connect to last table */
        }
        else
          *t = q;               /* first table is returned result */
      }

      /* set up table entry in r */
      r.bits = (Byte)(k - w);
      if (p >= v + n)
        r.exop = 128 + 64;      /* out of values--invalid code */
      else if (*p < s)
      {
        r.exop = (Byte)(*p < 256 ? 0 : 32 + 64);     /* 256 is end-of-block */
        r.base = *p++;          /* simple code is just the value */
      }
      else
      {
        r.exop = (Byte)(e[*p - s] + 16 + 64);/* non-simple--look up in lists */
        r.base = d[*p++ - s];
      }

      /* fill code-like entries with r */
      f = 1 << (k - w);
      for (j = i >> w; j < z; j += f)
        q[j] = r;

      /* backwards increment the k-bit code i */
      for (j = 1 << (k - 1); i & j; j >>= 1)
        i ^= j;
      i ^= j;

      /* backup over finished tables */
      mask = (1 << w) - 1;      /* needed on HP, cc -O bug */
      while ((i & mask) != x[h])
      {
        h--;                    /* don't need to update q */
        w -= l;
        mask = (1 << w) - 1;
      }
    }
  }


  /* Return Z_BUF_ERROR if we were given an incomplete table */
  return y != 0 && g != 1 ? Z_BUF_ERROR : Z_OK;
}
```

```
  }


  /* Find minimum and maximum length, bound *m by those */
  l = *m;
  for (j = 1; j <= BMAX; j++)
    if (c[j])
      break;
  k = j;                        /* minimum code length */
  if ((uInt)l < j)
    l = j;
  for (i = BMAX; i; i--)
    if (c[i])
      break;
  g = i;                        /* maximum code length */
  if ((uInt)l > i)
    l = i;
  *m = l;


  /* Adjust last length count to fill out codes, if needed */
  for (y = 1 << j; j < i; j++, y <<= 1)
    if ((y -= c[j]) < 0)
      return Z_DATA_ERROR;
  if ((y -= c[i]) < 0)
    return Z_DATA_ERROR;
  c[i] += y;


  /* Generate starting offsets into the value table for each length */
  x[1] = j = 0;
  p = c + 1;  xp = x + 2;
  while (--i) {                 /* note that i == g from above */
    *xp++ = (j += *p++);
  }


  /* Make a table of values in order of bit lengths */
  p = b;  i = 0;
  do {
    if ((j = *p++) != 0)
      v[x[j]++] = i;
  } while (++i < n);
  n = x[g];                     /* set n to length of v */


  /* Generate the Huffman codes and for each, make the table entries */
  x[0] = i = 0;                 /* first Huffman code is zero */
  p = v;                        /* grab values in bit order */
  h = -1;                       /* no tables yet--level -1 */
  w = -1;                       /* bits decoded == (l * h) */
  u[0] = (inflate_huft *)Z_NULL;        /* just to keep compilers happy */
  q = (inflate_huft *)Z_NULL;   /* ditto */
  z = 0;                        /* ditto */

  /* go through the bit lengths (k already is bits in shortest code) */
  for (; k <= g; k++)
  {
    a = c[k];
    while (a--)
    {
      /* here i is the Huffman code of length k bits for value *p */
      /* make tables up to required level */
      while (k > w + l)
      {
        h++;
        w += l;                 /* previous table always l bits */

        /* compute minimum size table less than or equal to l bits */
        z = g - w;
        z = z > (uInt)l ? l : z;        /* table size upper limit */
        if ((f = 1 << (j = k - w)) > a + 1)     /* try a k-w bit table */
        {                       /* too few codes for k-w bit table */
```

```
   codes are shorter than that, in which case the longest code length in
   bits is used, or when the shortest code is *longer* than the requested
   table size, in which case the length of the shortest code in bits is
   used.

   There are two different values for the two tables, since they code a
   different number of possibilities each.  The literal/length table
   codes 286 possible values, or in a flat code, a little over eight
   bits.  The distance table codes 30 possible values, or a little less
   than five bits, flat.  The optimum values for speed end up being
   about one bit more than those, so lbits is 8+1 and dbits is 5+1.
   The optimum values may differ though from machine to machine, and
   possibly even between compilers.  Your mileage may vary.
 */


/* If BMAX needs to be larger than 16, then h and x[] should be uLong. */
#define BMAX 15         /* maximum bit length of any code */

local int huft_build(b, n, s, d, e, t, m, hp, hn, v)
uIntf *b;               /* code lengths in bits (all assumed <= BMAX) */
uInt n;                 /* number of codes (assumed <= 288) */
uInt s;                 /* number of simple-valued codes (0..s-1) */
const uIntf *d;         /* list of base values for non-simple codes */
const uIntf *e;         /* list of extra bits for non-simple codes */
inflate_huft * FAR *t;  /* result: starting table */
uIntf *m;               /* maximum lookup bits, returns actual */
inflate_huft *hp;       /* space for trees */
uInt *hn;               /* hufts used in space */
uIntf *v;               /* working area: values in order of bit length */
/* Given a list of code lengths and a maximum table size, make a set of
   tables to decode that set of codes.  Return Z_OK on success, Z_BUF_ERROR
   if the given code set is incomplete (the tables are still built in this
   case), Z_DATA_ERROR if the input is invalid (an over-subscribed set of
   lengths), or Z_MEM_ERROR if not enough memory. */
{

  uInt a;                       /* counter for codes of length k */
  uInt c[BMAX+1];               /* bit length count table */
  uInt f;                       /* i repeats in table every f entries */
  int g;                        /* maximum code length */
  int h;                        /* table level */
  register uInt i;              /* counter, current code */
  register uInt j;              /* counter */
  register int k;               /* number of bits in current code */
  int l;                        /* bits per table (returned in m) */
  uInt mask;                    /* (1 << w) - 1, to avoid cc -O bug on HP */
  register uIntf *p;            /* pointer into c[], b[], or v[] */
  inflate_huft *q;              /* points to current table */
  struct inflate_huft_s r;      /* table entry for structure assignment */
  inflate_huft *u[BMAX];        /* table stack */
  register int w;               /* bits before this table == (l * h) */
  uInt x[BMAX+1];               /* bit offsets, then code stack */
  uIntf *xp;                    /* pointer into x */
  int y;                        /* number of dummy codes added */
  uInt z;                       /* number of entries in current table */


  /* Generate counts for each bit length */
  p = c;
#define C0 *p++ = 0;
#define C2 C0 C0 C0 C0
#define C4 C2 C2 C2 C2
  C4                            /* clear c[]--assume BMAX+1 is 16 */
  p = b;  i = n;
  do {
    c[*p++]++;                  /* assume all entries <= BMAX */
  } while (--i);
  if (c[0] == n)                /* null input--all zero length codes */
  {
    *t = (inflate_huft *)Z_NULL;
    *m = 0;
    return Z_OK;
```

```
/* inftrees.c -- generate Huffman trees for efficient decoding
 * Copyright (C) 1995-1998 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

#include "zutil.h"
#include "inftrees.h"

#if !defined(BUILDFIXED) && !defined(STDC)
#  define BUILDFIXED   /* non ANSI compilers may not accept inffixed.h */
#endif

const char inflate_copyright[] =
   " inflate 1.1.3 Copyright 1995-1998 Mark Adler ";
/*
  If you use the zlib library in a product, an acknowledgment is welcome
  in the documentation of your product. If for some reason you cannot
  include such an acknowledgment, I would appreciate that you keep this
  copyright string in the executable of your product.
 */
struct internal_state  {int dummy;}; /* for buggy compilers */

/* simplify the use of the inflate_huft type with some defines */
#define exop word.what.Exop
#define bits word.what.Bits


local int huft_build OF((
    uIntf *,            /* code lengths in bits */
    uInt,               /* number of codes */
    uInt,               /* number of "simple" codes */
    const uIntf *,      /* list of base values for non-simple codes */
    const uIntf *,      /* list of extra bits for non-simple codes */
    inflate_huft * FAR*,/* result: starting table */
    uIntf *,            /* maximum lookup bits (returns actual) */
    inflate_huft *,     /* space for trees */
    uInt *,             /* hufts used in space */
    uIntf * ));         /* space for values */

/* Tables for deflate from PKZIP's appnote.txt. */
local const uInt cplens[31] = { /* Copy lengths for literal codes 257..285 */
        3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 15, 17, 19, 23, 27, 31,
        35, 43, 51, 59, 67, 83, 99, 115, 131, 163, 195, 227, 258, 0, 0};
        /* see note #13 above about 258 */
local const uInt cplext[31] = { /* Extra bits for literal codes 257..285 */
        0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2,
        3, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 0, 112, 112}; /* 112==invalid */
local const uInt cpdist[30] = { /* Copy offsets for distance codes 0..29 */
        1, 2, 3, 4, 5, 7, 9, 13, 17, 25, 33, 49, 65, 97, 129, 193,
        257, 385, 513, 769, 1025, 1537, 2049, 3073, 4097, 6145,
        8193, 12289, 16385, 24577};
local const uInt cpdext[30] = { /* Extra bits for distance codes */
        0, 0, 0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6,
        7, 7, 8, 8, 9, 9, 10, 10, 11, 11,
        12, 12, 13, 13};

/*
   Huffman code decoding is performed using a multi-level table lookup.
   The fastest way to decode is to simply build a lookup table whose
   size is determined by the longest code.  However, the time it takes
   to build this table can also be a factor if the data being decoded
   is not very long.  The most common codes are necessarily the
   shortest codes, so those codes dominate the decoding time, and hence
   the speed.  The idea is you can have a shorter table that decodes the
   shorter, more probable codes, and then point to subsidiary tables for
   the longer codes.  The time it costs to decode the longer codes is
   then traded against the time it takes to make longer tables.

   This results of this trade are in the variables lbits and dbits
   below.  lbits is the number of bits the first level table for literal/
   length codes can decode in one step, and dbits is the same thing for
   the distance codes.  Subsequent tables are also less than or equal to
   those sizes.  These values may be adjusted either when all of the
```

```
/* inftrees.h -- header to use inftrees.c
 * Copyright (C) 1995-1998 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* WARNING: this file should *not* be used by applications. It is
   part of the implementation of the compression library and is
   subject to change. Applications should only use zlib.h.
 */

/* Huffman code lookup table entry--this entry is four bytes for machines
   that have 16-bit pointers (e.g. PC's in the small or medium model). */

typedef struct inflate_huft_s FAR inflate_huft;

struct inflate_huft_s {
  union {
    struct {
      Byte Exop;        /* number of extra bits or operation */
      Byte Bits;        /* number of bits in this code or subcode */
    } what;
    uInt pad;           /* pad structure to a power of 2 (4 bytes for */
  } word;               /*  16-bit, 8 bytes for 32-bit int's) */
  uInt base;            /* literal, length base, distance base,
                           or table offset */
};

/* Maximum size of dynamic tree.  The maximum found in a long but non-
   exhaustive search was 1004 huft structures (850 for length/literals
   and 154 for distances, the latter actually the result of an
   exhaustive search).  The actual maximum is not known, but the
   value below is more than safe. */
#define MANY 1440

extern int inflate_trees_bits OF((
    uIntf *,                    /* 19 code lengths */
    uIntf *,                    /* bits tree desired/actual depth */
    inflate_huft * FAR *,       /* bits tree result */
    inflate_huft *,             /* space for trees */
    z_streamp));                /* for messages */

extern int inflate_trees_dynamic OF((
    uInt,                       /* number of literal/length codes */
    uInt,                       /* number of distance codes */
    uIntf *,                    /* that many (total) code lengths */
    uIntf *,                    /* literal desired/actual bit depth */
    uIntf *,                    /* distance desired/actual bit depth */
    inflate_huft * FAR *,       /* literal/length tree result */
    inflate_huft * FAR *,       /* distance tree result */
    inflate_huft *,             /* space for trees */
    z_streamp));                /* for messages */

extern int inflate_trees_fixed OF((
    uIntf *,                    /* literal desired/actual bit depth */
    uIntf *,                    /* distance desired/actual bit depth */
    inflate_huft * FAR *,       /* literal/length tree result */
    inflate_huft * FAR *,       /* distance tree result */
    z_streamp));                /* for memory allocation */
    /* copy */
    zmemcpy(p, q, n);
    p += n;
    q += n;
  }

  /* update pointers */
  z->next_out = p;
  s->read = q;

  /* done */
  return r;
}
```

```
/* inflate_util.c -- data and routines common to blocks and codes
 * Copyright (C) 1995-1998 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

#include "zutil.h"
#include "infblock.h"
#include "inftrees.h"
#include "infcodes.h"
#include "infutil.h"

struct inflate_codes_state {int dummy;}; /* for buggy compilers */

/* And'ing with mask[n] masks the lower n bits */
uInt inflate_mask[17] = {
    0x0000,
    0x0001, 0x0003, 0x0007, 0x000f, 0x001f, 0x003f, 0x007f, 0x00ff,
    0x01ff, 0x03ff, 0x07ff, 0x0fff, 0x1fff, 0x3fff, 0x7fff, 0xffff
};


/* copy as much as possible from the sliding window to the output area */
int inflate_flush(s, z, r)
inflate_blocks_statef *s;
z_streamp z;
int r;
{
  uInt n;
  Bytef *p;
  Bytef *q;

  /* local copies of source and destination pointers */
  p = z->next_out;
  q = s->read;

  /* compute number of bytes to copy as far as end of window */
  n = (uInt)((q <= s->write ? s->write : s->end) - q);
  if (n > z->avail_out) n = z->avail_out;
  if (n && r == Z_BUF_ERROR) r = Z_OK;

  /* update counters */
  z->avail_out -= n;
  z->total_out += n;

  /* update check information */
  if (s->checkfn != Z_NULL)
    z->adler = s->check = (*s->checkfn)(s->check, q, n);

  /* copy as far as end of window */
  zmemcpy(p, q, n);
  p += n;
  q += n;

  /* see if more to copy at beginning of window */
  if (q == s->end)
  {
    /* wrap pointers */
    q = s->window;
    if (s->write == s->end)
      s->write = s->window;

    /* compute bytes to copy */
    n = (uInt)(s->write - q);
    if (n > z->avail_out) n = z->avail_out;
    if (n && r == Z_BUF_ERROR) r = Z_OK;

    /* update counters */
    z->avail_out -= n;
    z->total_out += n;

    /* update check information */
    if (s->checkfn != Z_NULL)
      z->adler = s->check = (*s->checkfn)(s->check, q, n);
```

```
/* infcodes.h -- header to use infcodes.c
 * Copyright (C) 1995-1998 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* WARNING: this file should *not* be used by applications. It is
   part of the implementation of the compression library and is
   subject to change. Applications should only use zlib.h.
 */

struct inflate_codes_state;
typedef struct inflate_codes_state FAR inflate_codes_statef;

extern inflate_codes_statef *inflate_codes_new OF((
    uInt, uInt,
    inflate_huft *, inflate_huft *,
    z_streamp ));

extern int inflate_codes OF((
    inflate_blocks_statef *,
    z_streamp ,
    int));

extern void inflate_codes_free OF((
    inflate_codes_statef *,
    z_streamp ));
```

```
/* infcodes.h -- header to use infcodes.c
 * Copyright (C) 1995-1998 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* WARNING: this file should *not* be used by applications. It is
   part of the implementation of the compression library and is
   subject to change. Applications should only use zlib.h.
 */

struct inflate_codes_state;
typedef struct inflate_codes_state FAR inflate_codes_statef;

extern inflate_codes_statef *inflate_codes_new OF((
    uInt, uInt,
    inflate_huft *, inflate_huft *,
    z_streamp ));

extern int inflate_codes OF((
    inflate_blocks_statef *,
    z_streamp ,
    int));

extern void inflate_codes_free OF((
    inflate_codes_statef *,
    z_streamp ));
```

```
#define NEXTBYTE (n--,*p++)
#define NEEDBITS(j) {while(k<(j)){NEEDBYTE;b|=((uLong)NEXTBYTE)<<k;k+=8;}}
#define DUMPBITS(j) {b>>=(j);k-=(j);}
/*   output bytes */
#define WAVAIL (uInt)(q<s->read?s->read-q-1:s->end-q)
#define LOADOUT {q=s->write;m=(uInt)WAVAIL;}
#define WRAP {if(q==s->end&&s->read!=s->window){q=s->window;m=(uInt)WAVAIL;}}
#define FLUSH {UPDOUT r=inflate_flush(s,z,r); LOADOUT}
#define NEEDOUT {if(m==0){WRAP if(m==0){FLUSH WRAP if(m==0) LEAVE}}r=Z_OK;}
#define OUTBYTE(a) {*q++=(Byte)(a);m--;}
/*   load local pointers */
#define LOAD {LOADIN LOADOUT}

/* masks for lower bits (size given to avoid silly warnings with Visual C++) */
extern uInt inflate_mask[17];

/* copy as much as possible from the sliding window to the output area */
extern int inflate_flush OF((
    inflate_blocks_statef *,
    z_streamp ,
    int));

struct internal_state      {int dummy;}; /* for buggy compilers */

#endif
```

09629370.073400

```
/* infutil.h -- types and macros common to blocks and codes
 * Copyright (C) 1995-1998 Mark Adler
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* WARNING: this file should *not* be used by applications. It is
   part of the implementation of the compression library and is
   subject to change. Applications should only use zlib.h.
 */

#ifndef _INFUTIL_H
#define _INFUTIL_H

typedef enum {
      TYPE,     /* get type bits (3, including end bit) */
      LENS,     /* get lengths for stored */
      STORED,   /* processing stored block */
      TABLE,    /* get table lengths */
      BTREE,    /* get bit lengths tree for a dynamic block */
      DTREE,    /* get length, distance trees for a dynamic block */
      CODES,    /* processing fixed or dynamic block */
      DRY,      /* output remaining window bytes */
      DONE,     /* finished last block, done */
      BAD}      /* got a data error--stuck here */
inflate_block_mode;

/* inflate blocks semi-private state */
struct inflate_blocks_state {

  /* mode */
  inflate_block_mode  mode;     /* current inflate_block mode */

  /* mode dependent information */
  union {
    uInt left;          /* if STORED, bytes left to copy */
    struct {
      uInt table;               /* table lengths (14 bits) */
      uInt index;               /* index into blens (or border) */
      uIntf *blens;             /* bit lengths of codes */
      uInt bb;                  /* bit length tree depth */
      inflate_huft *tb;         /* bit length decoding tree */
    } trees;            /* if DTREE, decoding info for trees */
    struct {
      inflate_codes_statef
         *codes;
    } decode;           /* if CODES, current state */
  } sub;                /* submode */
  uInt last;            /* true if this block is the last block */

  /* mode independent information */
  uInt bitk;            /* bits in bit buffer */
  uLong bitb;           /* bit buffer */
  inflate_huft *hufts;  /* single malloc for tree space */
  Bytef *window;        /* sliding window */
  Bytef *end;           /* one byte after sliding window */
  Bytef *read;          /* window read pointer */
  Bytef *write;         /* window write pointer */
  check_func checkfn;   /* check function */
  uLong check;          /* check on output */

};


/* defines for inflate input/output */
/*   update pointers and return */
#define UPDBITS {s->bitb=b;s->bitk=k;}
#define UPDIN {z->avail_in=n;z->total_in+=p-z->next_in;z->next_in=p;}
#define UPDOUT {s->write=q;}
#define UPDATE {UPDBITS UPDIN UPDOUT}
#define LEAVE {UPDATE return inflate_flush(s,z,r);}
/*   get bytes and bits */
#define LOADIN {p=z->next_in;n=z->avail_in;b=s->bitb;k=s->bitk;}
#define NEEDBYTE {if(n)r=Z_OK;else LEAVE}
```

```
/* ===========================================================================
 * Flush the bit buffer and align the output on a byte boundary
 */
local void bi_windup(s)
    deflate_state *s;
{
    if (s->bi_valid > 8) {
        put_short(s, s->bi_buf);
    } else if (s->bi_valid > 0) {
        put_byte(s, (Byte)s->bi_buf);
    }
    s->bi_buf = 0;
    s->bi_valid = 0;
#ifdef DEBUG
    s->bits_sent = (s->bits_sent+7) & ~7;
#endif
}

/* ===========================================================================
 * Copy a stored block, storing first the length and its
 * one's complement if requested.
 */
local void copy_block(s, buf, len, header)
    deflate_state *s;
    charf    *buf;    /* the input data */
    unsigned len;     /* its length */
    int      header;  /* true if block header must be written */
{
    bi_windup(s);        /* align on byte boundary */
    s->last_eob_len = 8; /* enough lookahead for inflate */

    if (header) {
        put_short(s, (ush)len);
        put_short(s, (ush)~len);
#ifdef DEBUG
        s->bits_sent += 2*16;
#endif
    }
#ifdef DEBUG
    s->bits_sent += (ulg)len<<3;
#endif
    while (len--) {
        put_byte(s, *buf++);
    }
}
```

```
            }
            dist--; /* dist is now the match distance - 1 */
            code = d_code(dist);
            Assert (code < D_CODES, "bad d_code");

            send_code(s, code, dtree);       /* send the distance code */
            extra = extra_dbits[code];
            if (extra != 0) {
                dist -= base_dist[code];
                send_bits(s, dist, extra);   /* send the extra distance bits */
            }
        } /* literal or match pair ? */

        /* Check that the overlay between pending_buf and d_buf+l_buf is ok: */
        Assert(s->pending < s->lit_bufsize + 2*lx, "pendingBuf overflow");

    } while (lx < s->last_lit);

    send_code(s, END_BLOCK, ltree);
    s->last_eob_len = ltree[END_BLOCK].Len;
}

/* ===========================================================================
 * Set the data type to ASCII or BINARY, using a crude approximation:
 * binary if more than 20% of the bytes are <= 6 or >= 128, ascii otherwise.
 * IN assertion: the fields freq of dyn_ltree are set and the total of all
 * frequencies does not exceed 64K (to fit in an int on 16 bit machines).
 */
local void set_data_type(s)
    deflate_state *s;
{
    int n = 0;
    unsigned ascii_freq = 0;
    unsigned bin_freq = 0;
    while (n < 7)        bin_freq += s->dyn_ltree[n++].Freq;
    while (n < 128)    ascii_freq += s->dyn_ltree[n++].Freq;
    while (n < LITERALS) bin_freq += s->dyn_ltree[n++].Freq;
    s->data_type = (Byte)(bin_freq > (ascii_freq >> 2) ? Z_BINARY : Z_ASCII);
}

/* ===========================================================================
 * Reverse the first len bits of a code, using straightforward code (a faster
 * method would use a table)
 * IN assertion: 1 <= len <= 15
 */
local unsigned bi_reverse(code, len)
    unsigned code; /* the value to invert */
    int len;       /* its bit length */
{
    register unsigned res = 0;
    do {
        res |= code & 1;
        code >>= 1, res <<= 1;
    } while (--len > 0);
    return res >> 1;
}

/* ===========================================================================
 * Flush the bit buffer, keeping at most 7 bits in it.
 */
local void bi_flush(s)
    deflate_state *s;
{
    if (s->bi_valid == 16) {
        put_short(s, s->bi_buf);
        s->bi_buf = 0;
        s->bi_valid = 0;
    } else if (s->bi_valid >= 8) {
        put_byte(s, (Byte)s->bi_buf);
        s->bi_buf >>= 8;
        s->bi_valid -= 8;
    }
}
```

```
    unsigned dist;  /* distance of matched string */
    unsigned lc;    /* match length-MIN_MATCH or unmatched char (if dist==0) */
{
    s->d_buf[s->last_lit] = (ush)dist;
    s->l_buf[s->last_lit++] = (uch)lc;
    if (dist == 0) {
        /* lc is the unmatched char */
        s->dyn_ltree[lc].Freq++;
    } else {
        s->matches++;
        /* Here, lc is the match length - MIN_MATCH */
        dist--;             /* dist = match distance - 1 */
        Assert((ush)dist < (ush)MAX_DIST(s) &&
               (ush)lc <= (ush)(MAX_MATCH-MIN_MATCH) &&
               (ush)d_code(dist) < (ush)D_CODES,  "_tr_tally: bad match");

        s->dyn_ltree[_length_code[lc]+LITERALS+1].Freq++;
        s->dyn_dtree[d_code(dist)].Freq++;
    }

#ifdef TRUNCATE_BLOCK
    /* Try to guess if it is profitable to stop the current block here */
    if ((s->last_lit & 0x1fff) == 0 && s->level > 2) {
        /* Compute an upper bound for the compressed length */
        ulg out_length = (ulg)s->last_lit*8L;
        ulg in_length = (ulg)((long)s->strstart - s->block_start);
        int dcode;
        for (dcode = 0; dcode < D_CODES; dcode++) {
            out_length += (ulg)s->dyn_dtree[dcode].Freq *
                (5L+extra_dbits[dcode]);
        }
        out_length >>= 3;
        Tracev((stderr,"\nlast_lit %u, in %ld, out ~%ld(%ld%%) ",
               s->last_lit, in_length, out_length,
               100L - out_length*100L/in_length));
        if (s->matches < s->last_lit/2 && out_length < in_length/2) return 1;
    }
#endif
    return (s->last_lit == s->lit_bufsize-1);
    /* We avoid equality with lit_bufsize because of wraparound at 64K
     * on 16 bit machines and because stored blocks are restricted to
     * 64K-1 bytes.
     */
}

/* ===========================================================================
 * Send the block data compressed using the given Huffman trees
 */
local void compress_block(s, ltree, dtree)
    deflate_state *s;
    ct_data *ltree; /* literal tree */
    ct_data *dtree; /* distance tree */
{
    unsigned dist;      /* distance of matched string */
    int lc;             /* match length or unmatched char (if dist == 0) */
    unsigned lx = 0;    /* running index in l_buf */
    unsigned code;      /* the code to send */
    int extra;          /* number of extra bits to send */

    if (s->last_lit != 0) do {
        dist = s->d_buf[lx];
        lc = s->l_buf[lx++];
        if (dist == 0) {
            send_code(s, lc, ltree); /* send a literal byte */
            Tracecv(isgraph(lc), (stderr," '%c' ", lc));
        } else {
            /* Here, lc is the match length - MIN_MATCH */
            code = _length_code[lc];
            send_code(s, code+LITERALS+1, ltree); /* send the length code */
            extra = extra_lbits[code];
            if (extra != 0) {
                lc -= base_length[code];
                send_bits(s, lc, extra);       /* send the extra length bits */
     */
    max_blindex = build_bl_tree(s);

    /* Determine the best encoding. Compute first the block length in bytes*/
    opt_lenb = (s->opt_len+3+7)>>3;
    static_lenb = (s->static_len+3+7)>>3;

    Tracev((stderr, "\nopt %lu(%lu) stat %lu(%lu) stored %lu lit %u ",
        opt_lenb, s->opt_len, static_lenb, s->static_len, stored_len,
        s->last_lit));

    if (static_lenb <= opt_lenb) opt_lenb = static_lenb;

    } else {
        Assert(buf != (char*)0, "lost buf");
    opt_lenb = static_lenb = stored_len + 5; /* force a stored block */
    }

#ifdef FORCE_STORED
    if (buf != (char*)0) { /* force stored block */
#else
    if (stored_len+4 <= opt_lenb && buf != (char*)0) {
                       /* 4: two words for the lengths */
#endif
        /* The test buf != NULL is only necessary if LIT_BUFSIZE > WSIZE.
         * Otherwise we can't have processed more than WSIZE input bytes since
         * the last block flush, because compression would have been
         * successful. If LIT_BUFSIZE <= WSIZE, it is never too late to
         * transform a block into a stored block.
         */
        _tr_stored_block(s, buf, stored_len, eof);

#ifdef FORCE_STATIC
    } else if (static_lenb >= 0) { /* force static trees */
#else
    } else if (static_lenb == opt_lenb) {
#endif
        send_bits(s, (STATIC_TREES<<1)+eof, 3);
        compress_block(s, (ct_data *)static_ltree, (ct_data *)static_dtree);
#ifdef DEBUG
        s->compressed_len += 3 + s->static_len;
#endif
    } else {
        send_bits(s, (DYN_TREES<<1)+eof, 3);
        send_all_trees(s, s->l_desc.max_code+1, s->d_desc.max_code+1,
                       max_blindex+1);
        compress_block(s, (ct_data *)s->dyn_ltree, (ct_data *)s->dyn_dtree);
#ifdef DEBUG
        s->compressed_len += 3 + s->opt_len;
#endif
    }
    Assert (s->compressed_len == s->bits_sent, "bad compressed size");
    /* The above check is made mod 2^32, for files larger than 512 MB
     * and uLong implemented on 32 bits.
     */
    init_block(s);

    if (eof) {
        bi_windup(s);
#ifdef DEBUG
        s->compressed_len += 7;  /* align on byte boundary */
#endif
    }
    Tracev((stderr,"\ncomprlen %lu(%lu) ", s->compressed_len>>3,
           s->compressed_len-7*eof));
}

/* ===========================================================================
 * Save the match info and tally the frequency counts. Return true if
 * the current block must be flushed.
 */
int _tr_tally (s, dist, lc)
    deflate_state *s;
```

```
#endif
    copy_block(s, buf, (unsigned)stored_len, 1); /* with header */
}

/* ===========================================================================
 * Send one empty static block to give enough lookahead for inflate.
 * This takes 10 bits, of which 7 may remain in the bit buffer.
 * The current inflate code requires 9 bits of lookahead. If the
 * last two codes for the previous block (real code plus EOB) were coded
 * on 5 bits or less, inflate may have only 5+3 bits of lookahead to decode
 * the last real code. In this case we send two empty static blocks instead
 * of one. (There are no problems if the previous block is stored or fixed.)
 * To simplify the code, we assume the worst case of last real code encoded
 * on one bit only.
 */
void _tr_align(s)
    deflate_state *s;
{
    send_bits(s, STATIC_TREES<<1, 3);
    send_code(s, END_BLOCK, static_ltree);
#ifdef DEBUG
    s->compressed_len += 10L; /* 3 for block type, 7 for EOB */
#endif
    bi_flush(s);
    /* Of the 10 bits for the empty block, we have already sent
     * (10 - bi_valid) bits. The lookahead for the last real code (before
     * the EOB of the previous block) was thus at least one plus the length
     * of the EOB plus what we have just sent of the empty static block.
     */
    if (1 + s->last_eob_len + 10 - s->bi_valid < 9) {
        send_bits(s, STATIC_TREES<<1, 3);
        send_code(s, END_BLOCK, static_ltree);
#ifdef DEBUG
        s->compressed_len += 10L;
#endif
        bi_flush(s);
    }
    s->last_eob_len = 7;
}

/* ===========================================================================
 * Determine the best encoding for the current block: dynamic trees, static
 * trees or store, and output the encoded block to the zip file.
 */
void _tr_flush_block(s, buf, stored_len, eof)
    deflate_state *s;
    charf *buf;       /* input block, or NULL if too old */
    ulg stored_len;   /* length of input block */
    int eof;          /* true if this is the last block for a file */
{
    ulg opt_lenb, static_lenb; /* opt_len and static_len in bytes */
    int max_blindex = 0;  /* index of last bit length code of non zero freq */

    /* Build the Huffman trees unless a stored block is forced */
    if (s->level > 0) {

     /* Check if the file is ascii or binary */
    if (s->data_type == Z_UNKNOWN) set_data_type(s);

    /* Construct the literal and distance trees */
    build_tree(s, (tree_desc *)(&(s->l_desc)));
    Tracev((stderr, "\nlit data: dyn %ld, stat %ld", s->opt_len,
        s->static_len));

    build_tree(s, (tree_desc *)(&(s->d_desc)));
    Tracev((stderr, "\ndist data: dyn %ld, stat %ld", s->opt_len,
        s->static_len));
    /* At this point, opt_len and static_len are the total bit lengths of
     * the compressed block data, excluding the tree representations.
     */

    /* Build the bit length tree for the above two trees, and get the index
     * in bl_order of the last bit length code to send.
    deflate_state *s;
{
    int max_blindex;  /* index of last bit length code of non zero freq */

    /* Determine the bit length frequencies for literal and distance trees */
    scan_tree(s, (ct_data *)s->dyn_ltree, s->l_desc.max_code);
    scan_tree(s, (ct_data *)s->dyn_dtree, s->d_desc.max_code);

    /* Build the bit length tree: */
    build_tree(s, (tree_desc *)(&(s->bl_desc)));
    /* opt_len now includes the length of the tree representations, except
     * the lengths of the bit lengths codes and the 5+5+4 bits for the counts.
     */

    /* Determine the number of bit length codes to send. The pkzip format
     * requires that at least 4 bit length codes be sent. (appnote.txt says
     * 3 but the actual value used is 4.)
     */
    for (max_blindex = BL_CODES-1; max_blindex >= 3; max_blindex--) {
        if (s->bl_tree[bl_order[max_blindex]].Len != 0) break;
    }
    /* Update opt_len to include the bit length tree and counts */
    s->opt_len += 3*(max_blindex+1) + 5+5+4;
    Tracev((stderr, "\ndyn trees: dyn %ld, stat %ld",
            s->opt_len, s->static_len));

    return max_blindex;
}

/* ===========================================================================
 * Send the header for a block using dynamic Huffman trees: the counts, the
 * lengths of the bit length codes, the literal tree and the distance tree.
 * IN assertion: lcodes >= 257, dcodes >= 1, blcodes >= 4.
 */
local void send_all_trees(s, lcodes, dcodes, blcodes)
    deflate_state *s;
    int lcodes, dcodes, blcodes; /* number of codes for each tree */
{
    int rank;                    /* index in bl_order */

    Assert (lcodes >= 257 && dcodes >= 1 && blcodes >= 4, "not enough codes");
    Assert (lcodes <= L_CODES && dcodes <= D_CODES && blcodes <= BL_CODES,
            "too many codes");
    Tracev((stderr, "\nbl counts: "));
    send_bits(s, lcodes-257, 5); /* not +255 as stated in appnote.txt */
    send_bits(s, dcodes-1,   5);
    send_bits(s, blcodes-4,  4); /* not -3 as stated in appnote.txt */
    for (rank = 0; rank < blcodes; rank++) {
        Tracev((stderr, "\nbl code %2d ", bl_order[rank]));
        send_bits(s, s->bl_tree[bl_order[rank]].Len, 3);
    }
    Tracev((stderr, "\nbl tree: sent %ld", s->bits_sent));

    send_tree(s, (ct_data *)s->dyn_ltree, lcodes-1); /* literal tree */
    Tracev((stderr, "\nlit tree: sent %ld", s->bits_sent));

    send_tree(s, (ct_data *)s->dyn_dtree, dcodes-1); /* distance tree */
    Tracev((stderr, "\ndist tree: sent %ld", s->bits_sent));
}

/* ===========================================================================
 * Send a stored block
 */
void _tr_stored_block(s, buf, stored_len, eof)
    deflate_state *s;
    charf *buf;       /* input block */
    ulg stored_len;   /* length of input block */
    int eof;          /* true if this is the last block for a file */
{
    send_bits(s, (STORED_BLOCK<<1)+eof, 3);  /* send block type */
#ifdef DEBUG
    s->compressed_len = (s->compressed_len + 3 + 7) & (ulg)~7L;
    s->compressed_len += (stored_len + 4) << 3;
```

```
            s->bl_tree[REP_3_6].Freq++;
        } else if (count <= 10) {
            s->bl_tree[REPZ_3_10].Freq++;
        } else {
            s->bl_tree[REPZ_11_138].Freq++;
        }
        count = 0; prevlen = curlen;
        if (nextlen == 0) {
            max_count = 138, min_count = 3;
        } else if (curlen == nextlen) {
            max_count = 6, min_count = 3;
        } else {
            max_count = 7, min_count = 4;
        }
    }
}

/* ===========================================================================
 * Send a literal or distance tree in compressed form, using the codes in
 * bl_tree.
 */
local void send_tree (s, tree, max_code)
    deflate_state *s;
    ct_data *tree; /* the tree to be scanned */
    int max_code;       /* and its largest code of non zero frequency */
{
    int n;                     /* iterates over all tree elements */
    int prevlen = -1;          /* last emitted length */
    int curlen;                /* length of current code */
    int nextlen = tree[0].Len; /* length of next code */
    int count = 0;             /* repeat count of the current code */
    int max_count = 7;         /* max repeat count */
    int min_count = 4;         /* min repeat count */

    /* tree[max_code+1].Len = -1; */  /* guard already set */
    if (nextlen == 0) max_count = 138, min_count = 3;

    for (n = 0; n <= max_code; n++) {
        curlen = nextlen; nextlen = tree[n+1].Len;
        if (++count < max_count && curlen == nextlen) {
            continue;
        } else if (count < min_count) {
            do { send_code(s, curlen, s->bl_tree); } while (--count != 0);

        } else if (curlen != 0) {
            if (curlen != prevlen) {
                send_code(s, curlen, s->bl_tree); count--;
            }
            Assert(count >= 3 && count <= 6, " 3_6?");
            send_code(s, REP_3_6, s->bl_tree); send_bits(s, count-3, 2);

        } else if (count <= 10) {
            send_code(s, REPZ_3_10, s->bl_tree); send_bits(s, count-3, 3);

        } else {
            send_code(s, REPZ_11_138, s->bl_tree); send_bits(s, count-11, 7);
        }
        count = 0; prevlen = curlen;
        if (nextlen == 0) {
            max_count = 138, min_count = 3;
        } else if (curlen == nextlen) {
            max_count = 6, min_count = 3;
        } else {
            max_count = 7, min_count = 4;
        }
    }
}

/* ===========================================================================
 * Construct the Huffman tree for the bit lengths and return the index in
 * bl_order of the last bit length code to send.
 */
local int build_bl_tree(s)
```

```c
    desc->max_code = max_code;

    /* The elements heap[heap_len/2+1 .. heap_len] are leaves of the tree,
     * establish sub-heaps of increasing lengths:
     */
    for (n = s->heap_len/2; n >= 1; n--) pqdownheap(s, tree, n);

    /* Construct the Huffman tree by repeatedly combining the least two
     * frequent nodes.
     */
    node = elems;              /* next internal node of the tree */
    do {
        pqremove(s, tree, n);  /* n = node of least frequency */
        m = s->heap[SMALLEST]; /* m = node of next least frequency */

        s->heap[--(s->heap_max)] = n; /* keep the nodes sorted by frequency */
        s->heap[--(s->heap_max)] = m;

        /* Create a new node father of n and m */
        tree[node].Freq = tree[n].Freq + tree[m].Freq;
        s->depth[node] = (uch) (MAX(s->depth[n], s->depth[m]) + 1);
        tree[n].Dad = tree[m].Dad = (ush)node;
#ifdef DUMP_BL_TREE
        if (tree == s->bl_tree) {
            fprintf(stderr,"\nnode %d(%d), sons %d(%d) %d(%d)",
                    node, tree[node].Freq, n, tree[n].Freq, m, tree[m].Freq);
        }
#endif
        /* and insert the new node in the heap */
        s->heap[SMALLEST] = node++;
        pqdownheap(s, tree, SMALLEST);

    } while (s->heap_len >= 2);

    s->heap[--(s->heap_max)] = s->heap[SMALLEST];

    /* At this point, the fields freq and dad are set. We can now
     * generate the bit lengths.
     */
    gen_bitlen(s, (tree_desc *)desc);

    /* The field len is now set, we can generate the bit codes */
    gen_codes ((ct_data *)tree, max_code, s->bl_count);
}

/* ===========================================================================
 * Scan a literal or distance tree to determine the frequencies of the codes
 * in the bit length tree.
 */
local void scan_tree (s, tree, max_code)
    deflate_state *s;
    ct_data *tree;   /* the tree to be scanned */
    int max_code;    /* and its largest code of non zero frequency */
{
    int n;                     /* iterates over all tree elements */
    int prevlen = -1;          /* last emitted length */
    int curlen;                /* length of current code */
    int nextlen = tree[0].Len; /* length of next code */
    int count = 0;             /* repeat count of the current code */
    int max_count = 7;         /* max repeat count */
    int min_count = 4;         /* min repeat count */

    if (nextlen == 0) max_count = 138, min_count = 3;
    tree[max_code+1].Len = (ush)0xffff; /* guard */

    for (n = 0; n <= max_code; n++) {
        curlen = nextlen; nextlen = tree[n+1].Len;
        if (++count < max_count && curlen == nextlen) {
            continue;
        } else if (count < min_count) {
            s->bl_tree[curlen].Freq += count;
        } else if (curlen != 0) {
            if (curlen != prevlen) s->bl_tree[curlen].Freq++;
    int bits;                   /* bit index */
    int n;                      /* code index */

    /* The distribution counts are first used to generate the code values
     * without bit reversal.
     */
    for (bits = 1; bits <= MAX_BITS; bits++) {
        next_code[bits] = code = (code + bl_count[bits-1]) << 1;
    }
    /* Check that the bit counts in bl_count are consistent. The last code
     * must be all ones.
     */
    Assert (code + bl_count[MAX_BITS]-1 == (1<<MAX_BITS)-1,
            "inconsistent bit counts");
    Tracev((stderr,"\ngen_codes: max_code %d ", max_code));

    for (n = 0;  n <= max_code; n++) {
        int len = tree[n].Len;
        if (len == 0) continue;
        /* Now reverse the bits */
        tree[n].Code = bi_reverse(next_code[len]++, len);

        Tracecv(tree != static_ltree, (stderr,"\nn %3d %c l %2d c %4x (%x) ",
             n, (isgraph(n) ? n : ' '), len, tree[n].Code, next_code[len]-1));
    }
}

/* ===========================================================================
 * Construct one Huffman tree and assigns the code bit strings and lengths.
 * Update the total bit length for the current block.
 * IN assertion: the field freq is set for all tree elements.
 * OUT assertions: the fields len and code are set to the optimal bit length
 *     and corresponding code. The length opt_len is updated; static_len is
 *     also updated if stree is not null. The field max_code is set.
 */
local void build_tree(s, desc)
    deflate_state *s;
    tree_desc *desc; /* the tree descriptor */
{
    ct_data *tree         = desc->dyn_tree;
    const ct_data *stree  = desc->stat_desc->static_tree;
    int elems             = desc->stat_desc->elems;
    int n, m;          /* iterate over heap elements */
    int max_code = -1; /* largest code with non zero frequency */
    int node;          /* new node being created */

    /* Construct the initial heap, with least frequent element in
     * heap[SMALLEST]. The sons of heap[n] are heap[2*n] and heap[2*n+1].
     * heap[0] is not used.
     */
    s->heap_len = 0, s->heap_max = HEAP_SIZE;

    for (n = 0; n < elems; n++) {
        if (tree[n].Freq != 0) {
            s->heap[++(s->heap_len)] = max_code = n;
            s->depth[n] = 0;
        } else {
            tree[n].Len = 0;
        }
    }

    /* The pkzip format requires that at least one distance code exists,
     * and that at least one bit should be sent even if there is only one
     * possible code. So to avoid special checks later on we force at least
     * two codes of non zero frequency.
     */
    while (s->heap_len < 2) {
        node = s->heap[++(s->heap_len)] = (max_code < 2 ? ++max_code : 0);
        tree[node].Freq = 1;
        s->depth[node] = 0;
        s->opt_len--; if (stree) s->static_len -= stree[node].Len;
        /* node is 0 or 1 so it does not have extra bits */
    }
```

```
     */
    tree[s->heap[s->heap_max]].Len = 0; /* root of the heap */

    for (h = s->heap_max+1; h < HEAP_SIZE; h++) {
        n = s->heap[h];
        bits = tree[tree[n].Dad].Len + 1;
        if (bits > max_length) bits = max_length, overflow++;
        tree[n].Len = (ush)bits;
        /* We overwrite tree[n].Dad which is no longer needed */

        if (n > max_code) continue; /* not a leaf node */

        s->bl_count[bits]++;
        xbits = 0;
        if (n >= base) xbits = extra[n-base];
        f = tree[n].Freq;
        s->opt_len += (ulg)f * (bits + xbits);
        if (stree) s->static_len += (ulg)f * (stree[n].Len + xbits);
    }
    if (overflow == 0) return;

    Trace((stderr,"\nbit length overflow\n"));
    /* This happens for example on obj2 and pic of the Calgary corpus */

    /* Find the first bit length which could increase: */
    do {
        bits = max_length-1;
        while (s->bl_count[bits] == 0) bits--;
        s->bl_count[bits]--;      /* move one leaf down the tree */
        s->bl_count[bits+1] += 2; /* move one overflow item as its brother */
        s->bl_count[max_length]--;
        /* The brother of the overflow item also moves one step up,
         * but this does not affect bl_count[max_length]
         */
        overflow -= 2;
    } while (overflow > 0);

    /* Now recompute all bit lengths, scanning in increasing frequency.
     * h is still equal to HEAP_SIZE. (It is simpler to reconstruct all
     * lengths instead of fixing only the wrong ones. This idea is taken
     * from 'ar' written by Haruhiko Okumura.)
     */
    for (bits = max_length; bits != 0; bits--) {
        n = s->bl_count[bits];
        while (n != 0) {
            m = s->heap[--h];
            if (m > max_code) continue;
            if (tree[m].Len != (unsigned) bits) {
                Trace((stderr,"code %d bits %d->%d\n", m, tree[m].Len, bits));
                s->opt_len += ((long)bits - (long)tree[m].Len)
                              *(long)tree[m].Freq;
                tree[m].Len = (ush)bits;
            }
            n--;
        }
    }
}

/* ===========================================================================
 * Generate the codes for a given tree and bit counts (which need not be
 * optimal).
 * IN assertion: the array bl_count contains the bit length statistics for
 * the given tree and the field len is set for all tree elements.
 * OUT assertion: the field code is set for all tree elements of non
 *     zero code length.
 */
local void gen_codes (tree, max_code, bl_count)
    ct_data *tree;             /* the tree to decorate */
    int max_code;              /* largest code with non zero frequency */
    ushf *bl_count;            /* number of codes at each bit length */
{
    ush next_code[MAX_BITS+1]; /* next code value for each bit length */
    ush code = 0;              /* running code value */
    pqdownheap(s, tree, SMALLEST); \
}

/* ===========================================================================
 * Compares to subtrees, using the tree depth as tie breaker when
 * the subtrees have equal frequency. This minimizes the worst case length.
 */
#define smaller(tree, n, m, depth) \
   (tree[n].Freq < tree[m].Freq || \
   (tree[n].Freq == tree[m].Freq && depth[n] <= depth[m]))

/* ===========================================================================
 * Restore the heap property by moving down the tree starting at node k,
 * exchanging a node with the smallest of its two sons if necessary, stopping
 * when the heap property is re-established (each father smaller than its
 * two sons).
 */
local void pqdownheap(s, tree, k)
    deflate_state *s;
    ct_data *tree;  /* the tree to restore */
    int k;               /* node to move down */
{
    int v = s->heap[k];
    int j = k << 1;  /* left son of k */
    while (j <= s->heap_len) {
        /* Set j to the smallest of the two sons: */
        if (j < s->heap_len &&
            smaller(tree, s->heap[j+1], s->heap[j], s->depth)) {
            j++;
        }
        /* Exit if v is smaller than both sons */
        if (smaller(tree, v, s->heap[j], s->depth)) break;

        /* Exchange v with the smallest son */
        s->heap[k] = s->heap[j];  k = j;

        /* And continue down the tree, setting j to the left son of k */
        j <<= 1;
    }
    s->heap[k] = v;
}

/* ===========================================================================
 * Compute the optimal bit lengths for a tree and update the total bit length
 * for the current block.
 * IN assertion: the fields freq and dad are set, heap[heap_max] and
 *    above are the tree nodes sorted by increasing frequency.
 * OUT assertions: the field len is set to the optimal bit length, the
 *     array bl_count contains the frequencies for each bit length.
 *     The length opt_len is updated; static_len is also updated if stree is
 *     not null.
 */
local void gen_bitlen(s, desc)
    deflate_state *s;
    tree_desc *desc;    /* the tree descriptor */
{
    ct_data *tree        = desc->dyn_tree;
    int max_code         = desc->max_code;
    const ct_data *stree = desc->stat_desc->static_tree;
    const intf *extra    = desc->stat_desc->extra_bits;
    int base             = desc->stat_desc->extra_base;
    int max_length       = desc->stat_desc->max_length;
    int h;              /* heap index */
    int n, m;           /* iterate over the tree elements */
    int bits;           /* bit length */
    int xbits;          /* extra bits */
    ush f;              /* frequency */
    int overflow = 0;   /* number of elements with bit length too large */

    for (bits = 0; bits <= MAX_BITS; bits++) s->bl_count[bits] = 0;

    /* In a first pass, compute the optimal bit lengths (which may
     * overflow in the case of the bit length tree).
    fprintf(header, "%1u%s", base_length[i],
        SEPARATOR(i, LENGTH_CODES-1, 20));
    }

    fprintf(header, "local const int base_dist[D_CODES] = {\n");
    for (i = 0; i < D_CODES; i++) {
    fprintf(header, "%5u%s", base_dist[i],
        SEPARATOR(i, D_CODES-1, 10));
    }

    fclose(header);
}
#endif /* GEN_TREES_H */

/* ===========================================================================
 * Initialize the tree data structures for a new zlib stream.
 */
void _tr_init(s)
    deflate_state *s;
{
    tr_static_init();

    s->l_desc.dyn_tree = s->dyn_ltree;
    s->l_desc.stat_desc = &static_l_desc;

    s->d_desc.dyn_tree = s->dyn_dtree;
    s->d_desc.stat_desc = &static_d_desc;

    s->bl_desc.dyn_tree = s->bl_tree;
    s->bl_desc.stat_desc = &static_bl_desc;

    s->bi_buf = 0;
    s->bi_valid = 0;
    s->last_eob_len = 8; /* enough lookahead for inflate */
#ifdef DEBUG
    s->compressed_len = 0L;
    s->bits_sent = 0L;
#endif

    /* Initialize the first block of the first file: */
    init_block(s);
}

/* ===========================================================================
 * Initialize a new block.
 */
local void init_block(s)
    deflate_state *s;
{
    int n; /* iterates over tree elements */

    /* Initialize the trees. */
    for (n = 0; n < L_CODES;  n++) s->dyn_ltree[n].Freq = 0;
    for (n = 0; n < D_CODES;  n++) s->dyn_dtree[n].Freq = 0;
    for (n = 0; n < BL_CODES; n++) s->bl_tree[n].Freq = 0;

    s->dyn_ltree[END_BLOCK].Freq = 1;
    s->opt_len = s->static_len = 0L;
    s->last_lit = s->matches = 0;
}

#define SMALLEST 1
/* Index within the heap array of least frequent node in the Huffman tree */


/* ===========================================================================
 * Remove the smallest element from the heap and recreate the heap with
 * one less element. Updates heap and heap_len.
 */
#define pqremove(s, tree, top) \
{\
    top = s->heap[SMALLEST]; \
    s->heap[SMALLEST] = s->heap[s->heap_len--]; \
    /* Construct the codes of the static literal tree */
    for (bits = 0; bits <= MAX_BITS; bits++) bl_count[bits] = 0;
    n = 0;
    while (n <= 143) static_ltree[n++].Len = 8, bl_count[8]++;
    while (n <= 255) static_ltree[n++].Len = 9, bl_count[9]++;
    while (n <= 279) static_ltree[n++].Len = 7, bl_count[7]++;
    while (n <= 287) static_ltree[n++].Len = 8, bl_count[8]++;
    /* Codes 286 and 287 do not exist, but we must include them in the
     * tree construction to get a canonical Huffman tree (longest code
     * all ones)
     */
    gen_codes((ct_data *)static_ltree, L_CODES+1, bl_count);

    /* The static distance tree is trivial: */
    for (n = 0; n < D_CODES; n++) {
        static_dtree[n].Len = 5;
        static_dtree[n].Code = bi_reverse((unsigned)n, 5);
    }
    static_init_done = 1;

#  ifdef GEN_TREES_H
    gen_trees_header();
#  endif
#endif /* defined(GEN_TREES_H) || !defined(STDC) */
}

/* ===========================================================================
 * Genererate the file trees.h describing the static trees.
 */
#ifdef GEN_TREES_H
#  ifndef DEBUG
#   include <stdio.h>
#  endif

#  define SEPARATOR(i, last, width) \
      ((i) == (last)? "\n};\n\n" :    \
       ((i) % (width) == (width)-1 ? ",\n" : ", "))

void gen_trees_header()
{
    FILE *header = fopen("trees.h", "w");
    int i;

    Assert (header != NULL, "Can't open trees.h");
    fprintf(header,
        "/* header created automatically with -DGEN_TREES_H */\n\n");

    fprintf(header, "local const ct_data static_ltree[L_CODES+2] = {\n");
    for (i = 0; i < L_CODES+2; i++) {
    fprintf(header, "{{%3u},{%3u}}%s", static_ltree[i].Code,
        static_ltree[i].Len, SEPARATOR(i, L_CODES+1, 5));
    }

    fprintf(header, "local const ct_data static_dtree[D_CODES] = {\n");
    for (i = 0; i < D_CODES; i++) {
    fprintf(header, "{{%2u},{%2u}}%s", static_dtree[i].Code,
        static_dtree[i].Len, SEPARATOR(i, D_CODES-1, 5));
    }

    fprintf(header, "const uch _dist_code[DIST_CODE_LEN] = {\n");
    for (i = 0; i < DIST_CODE_LEN; i++) {
    fprintf(header, "%2u%s", _dist_code[i],
        SEPARATOR(i, DIST_CODE_LEN-1, 20));
    }

    fprintf(header, "const uch _length_code[MAX_MATCH-MIN_MATCH+1]= {\n");
    for (i = 0; i < MAX_MATCH-MIN_MATCH+1; i++) {
    fprintf(header, "%2u%s", _length_code[i],
        SEPARATOR(i, MAX_MATCH-MIN_MATCH, 20));
    }

    fprintf(header, "local const int base_length[LENGTH_CODES] = {\n");
    for (i = 0; i < LENGTH_CODES; i++) {
```

```
    int val = value;\
    s->bi_buf |= (val << s->bi_valid);\
    put_short(s, s->bi_buf);\
    s->bi_buf = (ush)val >> (Buf_size - s->bi_valid);\
    s->bi_valid += len - Buf_size;\
  } else {\
    s->bi_buf |= (value) << s->bi_valid;\
    s->bi_valid += len;\
  }\
}
#endif /* DEBUG */


#define MAX(a,b) (a >= b ? a : b)
/* the arguments must not have side effects */

/* ===========================================================================
 * Initialize the various 'constant' tables.
 */
local void tr_static_init()
{
#if defined(GEN_TREES_H) || !defined(STDC)
    static int static_init_done = 0;
    int n;        /* iterates over tree elements */
    int bits;     /* bit counter */
    int length;   /* length value */
    int code;     /* code value */
    int dist;     /* distance index */
    ush bl_count[MAX_BITS+1];
    /* number of codes at each bit length for an optimal tree */

    if (static_init_done) return;

    /* For some embedded targets, global variables are not initialized: */
    static_l_desc.static_tree = static_ltree;
    static_l_desc.extra_bits = extra_lbits;
    static_d_desc.static_tree = static_dtree;
    static_d_desc.extra_bits = extra_dbits;
    static_bl_desc.extra_bits = extra_blbits;

    /* Initialize the mapping length (0..255) -> length code (0..28) */
    length = 0;
    for (code = 0; code < LENGTH_CODES-1; code++) {
        base_length[code] = length;
        for (n = 0; n < (1<<extra_lbits[code]); n++) {
            _length_code[length++] = (uch)code;
        }
    }
    Assert (length == 256, "tr_static_init: length != 256");
    /* Note that the length 255 (match length 258) can be represented
     * in two different ways: code 284 + 5 bits or code 285, so we
     * overwrite length_code[255] to use the best encoding:
     */
    _length_code[length-1] = (uch)code;

    /* Initialize the mapping dist (0..32K) -> dist code (0..29) */
    dist = 0;
    for (code = 0 ; code < 16; code++) {
        base_dist[code] = dist;
        for (n = 0; n < (1<<extra_dbits[code]); n++) {
            _dist_code[dist++] = (uch)code;
        }
    }
    Assert (dist == 256, "tr_static_init: dist != 256");
    dist >>= 7; /* from now on, all distances are divided by 128 */
    for ( ; code < D_CODES; code++) {
        base_dist[code] = dist << 7;
        for (n = 0; n < (1<<(extra_dbits[code]-7)); n++) {
            _dist_code[256 + dist++] = (uch)code;
        }
    }
    Assert (dist == 256, "tr_static_init: 256+dist != 512");
```

```
local void build_tree     OF((deflate_state *s, tree_desc *desc));
local void scan_tree      OF((deflate_state *s, ct_data *tree, int max_code));
local void send_tree      OF((deflate_state *s, ct_data *tree, int max_code));
local int  build_bl_tree  OF((deflate_state *s));
local void send_all_trees OF((deflate_state *s, int lcodes, int dcodes,
                              int blcodes));
local void compress_block OF((deflate_state *s, ct_data *ltree,
                              ct_data *dtree));
local void set_data_type  OF((deflate_state *s));
local unsigned bi_reverse OF((unsigned value, int length));
local void bi_windup      OF((deflate_state *s));
local void bi_flush       OF((deflate_state *s));
local void copy_block     OF((deflate_state *s, charf *buf, unsigned len,
                              int header));

#ifdef GEN_TREES_H
local void gen_trees_header OF((void));
#endif

#ifndef DEBUG
#  define send_code(s, c, tree) send_bits(s, tree[c].Code, tree[c].Len)
   /* Send a code of the given tree. c and tree must not have side effects */

#else /* DEBUG */
#  define send_code(s, c, tree) \
     { if (z_verbose>2) fprintf(stderr,"\ncd %3d ",(c)); \
       send_bits(s, tree[c].Code, tree[c].Len); }
#endif

/* ===========================================================================
 * Output a short LSB first on the stream.
 * IN assertion: there is enough room in pendingBuf.
 */
#define put_short(s, w) { \
    put_byte(s, (uch)((w) & 0xff)); \
    put_byte(s, (uch)((ush)(w) >> 8)); \
}

/* ===========================================================================
 * Send a value on a given number of bits.
 * IN assertion: length <= 16 and value fits in length bits.
 */
#ifdef DEBUG
local void send_bits      OF((deflate_state *s, int value, int length));

local void send_bits(s, value, length)
    deflate_state *s;
    int value;  /* value to send */
    int length; /* number of bits */
{
    Tracevv((stderr," l %2d v %4x ", length, value));
    Assert(length > 0 && length <= 15, "invalid length");
    s->bits_sent += (ulg)length;

    /* If not enough room in bi_buf, use (valid) bits from bi_buf and
     * (16 - bi_valid) bits from value, leaving (width - (16-bi_valid))
     * unused bits in value.
     */
    if (s->bi_valid > (int)Buf_size - length) {
        s->bi_buf |= (value << s->bi_valid);
        put_short(s, s->bi_buf);
        s->bi_buf = (ush)value >> (Buf_size - s->bi_valid);
        s->bi_valid += length - Buf_size;
    } else {
        s->bi_buf |= value << s->bi_valid;
        s->bi_valid += length;
    }
}
#else /* !DEBUG */

#define send_bits(s, value, length) \
{ int len = length;\
  if (s->bi_valid > (int)Buf_size - len) {\
 */

#define Buf_size (8 * 2*sizeof(char))
/* Number of bits used within bi_buf. (bi_buf might be implemented on
 * more than 16 bits on some systems.)
 */

/* ===========================================================================
 * Local data. These are initialized only once.
 */

#define DIST_CODE_LEN  512 /* see definition of array dist_code below */

#if defined(GEN_TREES_H) || !defined(STDC)
/* non ANSI compilers may not accept trees.h */

local ct_data static_ltree[L_CODES+2];
/* The static literal tree. Since the bit lengths are imposed, there is no
 * need for the L_CODES extra codes used during heap construction. However
 * The codes 286 and 287 are needed to build a canonical tree (see _tr_init
 * below).
 */

local ct_data static_dtree[D_CODES];
/* The static distance tree. (Actually a trivial tree since all codes use
 * 5 bits.)
 */

uch _dist_code[DIST_CODE_LEN];
/* Distance codes. The first 256 values correspond to the distances
 * 3 .. 258, the last 256 values correspond to the top 8 bits of
 * the 15 bit distances.
 */

uch _length_code[MAX_MATCH-MIN_MATCH+1];
/* length code for each normalized match length (0 == MIN_MATCH) */

local int base_length[LENGTH_CODES];
/* First normalized length for each code (0 = MIN_MATCH) */

local int base_dist[D_CODES];
/* First normalized distance for each code (0 = distance of 1) */

#else
#  include "trees.h"
#endif /* GEN_TREES_H */

struct static_tree_desc_s {
    const ct_data *static_tree;  /* static tree or NULL */
    const intf *extra_bits;      /* extra bits for each code or NULL */
    int     extra_base;          /* base index for extra_bits */
    int     elems;               /* max number of elements in the tree */
    int     max_length;          /* max bit length for the codes */
};

local static_tree_desc  static_l_desc =
{static_ltree, extra_lbits, LITERALS+1, L_CODES, MAX_BITS};

local static_tree_desc  static_d_desc =
{static_dtree, extra_dbits, 0,          D_CODES, MAX_BITS};

local static_tree_desc  static_bl_desc =
{(const ct_data *)0, extra_blbits, 0,   BL_CODES, MAX_BL_BITS};

/* ===========================================================================
 * Local (static) routines in this file.
 */

local void tr_static_init OF((void));
local void init_block     OF((deflate_state *s));
local void pqdownheap     OF((deflate_state *s, ct_data *tree, int k));
local void gen_bitlen     OF((deflate_state *s, tree_desc *desc));
local void gen_codes      OF((ct_data *tree, int max_code, ushf *bl_count));
```

```
/* trees.c -- output deflated data using Huffman coding
 * Copyright (C) 1995-1998 Jean-loup Gailly
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/*
 *  ALGORITHM
 *
 *      The "deflation" process uses several Huffman trees. The more
 *      common source values are represented by shorter bit sequences.
 *
 *      Each code tree is stored in a compressed form which is itself
 * a Huffman encoding of the lengths of all the code strings (in
 * ascending order by source values).  The actual code strings are
 * reconstructed from the lengths in the inflate process, as described
 * in the deflate specification.
 *
 *  REFERENCES
 *
 *      Deutsch, L.P.,"'Deflate' Compressed Data Format Specification".
 *      Available in ftp.uu.net:/pub/archiving/zip/doc/deflate-1.1.doc
 *
 *      Storer, James A.
 *          Data Compression:  Methods and Theory, pp. 49-50.
 *          Computer Science Press, 1988.  ISBN 0-7167-8156-5.
 *
 *      Sedgewick, R.
 *          Algorithms, p290.
 *          Addison-Wesley, 1983. ISBN 0-201-06672-6.
 */

/* @(#) $Id$ */

/* #define GEN_TREES_H */

#include "deflate.h"

#ifdef DEBUG
#  include <ctype.h>
#endif

/* ===========================================================================
 * Constants
 */

#define MAX_BL_BITS 7
/* Bit length codes must not exceed MAX_BL_BITS bits */

#define END_BLOCK 256
/* end of block literal code */

#define REP_3_6      16
/* repeat previous bit length 3-6 times (2 bits of repeat count) */

#define REPZ_3_10    17
/* repeat a zero length 3-10 times  (3 bits of repeat count) */

#define REPZ_11_138  18
/* repeat a zero length 11-138 times  (7 bits of repeat count) */

local const int extra_lbits[LENGTH_CODES] /* extra bits for each length code */
   = {0,0,0,0,0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5,0};

local const int extra_dbits[D_CODES] /* extra bits for each distance code */
   = {0,0,0,0,1,1,2,2,3,3,4,4,5,5,6,6,7,7,8,8,9,9,10,10,11,11,12,12,13,13};

local const int extra_blbits[BL_CODES]/* extra bits for each bit length code */
   = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,3,7};

local const uch bl_order[BL_CODES]
   = {16,17,18,0,8,7,9,6,10,5,11,4,12,3,13,2,14,1,15};
/* The lengths of the bit length codes are sent in order of decreasing
 * probability, to avoid transmitting the lengths for unused bit length codes.
 0,  1,  2,  3,  4,  4,  5,  5,  6,  6,  6,  6,  7,  7,  7,  7,  8,  8,  8,  8,
 8,  8,  8,  8,  9,  9,  9,  9,  9,  9,  9,  9, 10, 10, 10, 10, 10, 10, 10, 10,
10, 10, 10, 10, 10, 10, 10, 10, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11,
11, 11, 11, 11, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12,
12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 13, 13, 13, 13,
13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13,
13, 13, 13, 13, 13, 13, 13, 13, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14,
14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14,
14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14,
14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 15, 15, 15, 15, 15, 15, 15, 15,
15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15,
15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15,
15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15,  0,  0, 16, 17,
18, 18, 19, 19, 20, 20, 20, 20, 21, 21, 21, 21, 22, 22, 22, 22, 22, 22, 22, 22,
23, 23, 23, 23, 23, 23, 23, 23, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24,
24, 24, 24, 24, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,
26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26,
26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 27, 27, 27, 27, 27, 27, 27, 27,
27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27,
27, 27, 27, 27, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28,
28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28,
28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28,
28, 28, 28, 28, 28, 28, 28, 28, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29,
29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29,
29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29,
29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29
};

const uch _length_code[MAX_MATCH-MIN_MATCH+1]= {
 0,  1,  2,  3,  4,  5,  6,  7,  8,  8,  9,  9, 10, 10, 11, 11, 12, 12, 12, 12,
13, 13, 13, 13, 14, 14, 14, 14, 15, 15, 15, 15, 16, 16, 16, 16, 16, 16, 16, 16,
17, 17, 17, 17, 17, 17, 17, 17, 18, 18, 18, 18, 18, 18, 18, 18, 19, 19, 19, 19,
19, 19, 19, 19, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 22, 22, 22, 22,
22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 23, 23, 23, 23, 23, 23, 23, 23,
23, 23, 23, 23, 23, 23, 23, 23, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24,
24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24,
25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,
25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 26, 26, 26, 26, 26, 26, 26, 26,
26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 26,
26, 26, 26, 26, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27,
27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 28
};

local const int base_length[LENGTH_CODES] = {
0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 16, 20, 24, 28, 32, 40, 48, 56,
64, 80, 96, 112, 128, 160, 192, 224, 0
};

local const int base_dist[D_CODES] = {
    0,     1,     2,     3,     4,     6,     8,    12,    16,    24,
   32,    48,    64,    96,   128,   192,   256,   384,   512,   768,
 1024,  1536,  2048,  3072,  4096,  6144,  8192, 12288, 16384, 24576
};
```

```
/* header created automatically with -DGEN_TREES_H */

local const ct_data static_ltree[L_CODES+2] = {
{{ 12},{  8}}, {{140},{  8}}, {{ 76},{  8}}, {{204},{  8}}, {{ 44},{  8}},
{{172},{  8}}, {{108},{  8}}, {{236},{  8}}, {{ 28},{  8}}, {{156},{  8}},
{{ 92},{  8}}, {{220},{  8}}, {{ 60},{  8}}, {{188},{  8}}, {{124},{  8}},
{{252},{  8}}, {{  2},{  8}}, {{130},{  8}}, {{ 66},{  8}}, {{194},{  8}},
{{ 34},{  8}}, {{162},{  8}}, {{ 98},{  8}}, {{226},{  8}}, {{ 18},{  8}},
{{146},{  8}}, {{ 82},{  8}}, {{210},{  8}}, {{ 50},{  8}}, {{178},{  8}},
{{114},{  8}}, {{242},{  8}}, {{ 10},{  8}}, {{138},{  8}}, {{ 74},{  8}},
{{202},{  8}}, {{ 42},{  8}}, {{170},{  8}}, {{106},{  8}}, {{234},{  8}},
{{ 26},{  8}}, {{154},{  8}}, {{ 90},{  8}}, {{218},{  8}}, {{ 58},{  8}},
{{186},{  8}}, {{122},{  8}}, {{250},{  8}}, {{  6},{  8}}, {{134},{  8}},
{{ 70},{  8}}, {{198},{  8}}, {{ 38},{  8}}, {{166},{  8}}, {{102},{  8}},
{{230},{  8}}, {{ 22},{  8}}, {{150},{  8}}, {{ 86},{  8}}, {{214},{  8}},
{{ 54},{  8}}, {{182},{  8}}, {{118},{  8}}, {{246},{  8}}, {{ 14},{  8}},
{{142},{  8}}, {{ 78},{  8}}, {{206},{  8}}, {{ 46},{  8}}, {{174},{  8}},
{{110},{  8}}, {{238},{  8}}, {{ 30},{  8}}, {{158},{  8}}, {{ 94},{  8}},
{{222},{  8}}, {{ 62},{  8}}, {{190},{  8}}, {{126},{  8}}, {{254},{  8}},
{{  1},{  8}}, {{129},{  8}}, {{ 65},{  8}}, {{193},{  8}}, {{ 33},{  8}},
{{161},{  8}}, {{ 97},{  8}}, {{225},{  8}}, {{ 17},{  8}}, {{145},{  8}},
{{ 81},{  8}}, {{209},{  8}}, {{ 49},{  8}}, {{177},{  8}}, {{113},{  8}},
{{241},{  8}}, {{  9},{  8}}, {{137},{  8}}, {{ 73},{  8}}, {{201},{  8}},
{{ 41},{  8}}, {{169},{  8}}, {{105},{  8}}, {{233},{  8}}, {{ 25},{  8}},
{{153},{  8}}, {{ 89},{  8}}, {{217},{  8}}, {{ 57},{  8}}, {{185},{  8}},
{{121},{  8}}, {{249},{  8}}, {{  5},{  8}}, {{133},{  8}}, {{ 69},{  8}},
{{197},{  8}}, {{ 37},{  8}}, {{165},{  8}}, {{101},{  8}}, {{229},{  8}},
{{ 21},{  8}}, {{149},{  8}}, {{ 85},{  8}}, {{213},{  8}}, {{ 53},{  8}},
{{181},{  8}}, {{117},{  8}}, {{245},{  8}}, {{ 13},{  8}}, {{141},{  8}},
{{ 77},{  8}}, {{205},{  8}}, {{ 45},{  8}}, {{173},{  8}}, {{109},{  8}},
{{237},{  8}}, {{ 29},{  8}}, {{157},{  8}}, {{ 93},{  8}}, {{221},{  8}},
{{ 61},{  8}}, {{189},{  8}}, {{125},{  8}}, {{253},{  8}}, {{ 19},{  9}},
{{275},{  9}}, {{147},{  9}}, {{403},{  9}}, {{ 83},{  9}}, {{339},{  9}},
{{211},{  9}}, {{467},{  9}}, {{ 51},{  9}}, {{307},{  9}}, {{179},{  9}},
{{435},{  9}}, {{115},{  9}}, {{371},{  9}}, {{243},{  9}}, {{499},{  9}},
{{ 11},{  9}}, {{267},{  9}}, {{139},{  9}}, {{395},{  9}}, {{ 75},{  9}},
{{331},{  9}}, {{203},{  9}}, {{459},{  9}}, {{ 43},{  9}}, {{299},{  9}},
{{171},{  9}}, {{427},{  9}}, {{107},{  9}}, {{363},{  9}}, {{235},{  9}},
{{491},{  9}}, {{ 27},{  9}}, {{283},{  9}}, {{155},{  9}}, {{411},{  9}},
{{ 91},{  9}}, {{347},{  9}}, {{219},{  9}}, {{475},{  9}}, {{ 59},{  9}},
{{315},{  9}}, {{187},{  9}}, {{443},{  9}}, {{123},{  9}}, {{379},{  9}},
{{251},{  9}}, {{507},{  9}}, {{  7},{  9}}, {{263},{  9}}, {{135},{  9}},
{{391},{  9}}, {{ 71},{  9}}, {{327},{  9}}, {{199},{  9}}, {{455},{  9}},
{{ 39},{  9}}, {{295},{  9}}, {{167},{  9}}, {{423},{  9}}, {{103},{  9}},
{{359},{  9}}, {{231},{  9}}, {{487},{  9}}, {{ 23},{  9}}, {{279},{  9}},
{{151},{  9}}, {{407},{  9}}, {{ 87},{  9}}, {{343},{  9}}, {{215},{  9}},
{{471},{  9}}, {{ 55},{  9}}, {{311},{  9}}, {{183},{  9}}, {{439},{  9}},
{{119},{  9}}, {{375},{  9}}, {{247},{  9}}, {{503},{  9}}, {{ 15},{  9}},
{{271},{  9}}, {{143},{  9}}, {{399},{  9}}, {{ 79},{  9}}, {{335},{  9}},
{{207},{  9}}, {{463},{  9}}, {{ 47},{  9}}, {{303},{  9}}, {{175},{  9}},
{{431},{  9}}, {{111},{  9}}, {{367},{  9}}, {{239},{  9}}, {{495},{  9}},
{{ 31},{  9}}, {{287},{  9}}, {{159},{  9}}, {{415},{  9}}, {{ 95},{  9}},
{{351},{  9}}, {{223},{  9}}, {{479},{  9}}, {{ 63},{  9}}, {{319},{  9}},
{{191},{  9}}, {{447},{  9}}, {{127},{  9}}, {{383},{  9}}, {{255},{  9}},
{{511},{  9}}, {{  0},{  7}}, {{ 64},{  7}}, {{ 32},{  7}}, {{ 96},{  7}},
{{ 16},{  7}}, {{ 80},{  7}}, {{ 48},{  7}}, {{112},{  7}}, {{  8},{  7}},
{{ 72},{  7}}, {{ 40},{  7}}, {{104},{  7}}, {{ 24},{  7}}, {{ 88},{  7}},
{{ 56},{  7}}, {{120},{  7}}, {{  4},{  7}}, {{ 68},{  7}}, {{ 36},{  7}},
{{100},{  7}}, {{ 20},{  7}}, {{ 84},{  7}}, {{ 52},{  7}}, {{116},{  7}},
{{  3},{  8}}, {{131},{  8}}, {{ 67},{  8}}, {{195},{  8}}, {{ 35},{  8}},
{{163},{  8}}, {{ 99},{  8}}, {{227},{  8}}
};

local const ct_data static_dtree[D_CODES] = {
{{ 0},{ 5}}, {{16},{ 5}}, {{ 8},{ 5}}, {{24},{ 5}}, {{ 4},{ 5}},
{{20},{ 5}}, {{12},{ 5}}, {{28},{ 5}}, {{ 2},{ 5}}, {{18},{ 5}},
{{10},{ 5}}, {{26},{ 5}}, {{ 6},{ 5}}, {{22},{ 5}}, {{14},{ 5}},
{{30},{ 5}}, {{ 1},{ 5}}, {{17},{ 5}}, {{ 9},{ 5}}, {{25},{ 5}},
{{ 5},{ 5}}, {{21},{ 5}}, {{13},{ 5}}, {{29},{ 5}}, {{ 3},{ 5}},
{{19},{ 5}}, {{11},{ 5}}, {{27},{ 5}}, {{ 7},{ 5}}, {{23},{ 5}}
};

const uch _dist_code[DIST_CODE_LEN] = {
```

File: Work\CrtPrt\zcomp\uncompr[redacted]

```c
/* uncompr.c -- decompress a memory buffer
 * Copyright (C) 1995-1998 Jean-loup Gailly.
 * For conditions of distribution and use, see copyright notice in zlib.h
 */

/* @(#) $Id$ */

#include "zlib.h"

/* ===========================================================================
     Decompresses the source buffer into the destination buffer.  sourceLen is
   the byte length of the source buffer. Upon entry, destLen is the total
   size of the destination buffer, which must be large enough to hold the
   entire uncompressed data. (The size of the uncompressed data must have
   been saved previously by the compressor and transmitted to the decompressor
   by some mechanism outside the scope of this compression library.)
   Upon exit, destLen is the actual size of the compressed buffer.
     This function can be used to decompress a whole file at once if the
   input file is mmap'ed.

     uncompress returns Z_OK if success, Z_MEM_ERROR if there was not
   enough memory, Z_BUF_ERROR if there was not enough room in the output
   buffer, or Z_DATA_ERROR if the input data was corrupted.
*/
int ZEXPORT uncompress (dest, destLen, source, sourceLen)
    Bytef *dest;
    uLongf *destLen;
    const Bytef *source;
    uLong sourceLen;
{
    z_stream stream;
    int err;

    stream.next_in = (Bytef*)source;
    stream.avail_in = (uInt)sourceLen;
    /* Check for source > 64K on 16-bit machine: */
    if ((uLong)stream.avail_in != sourceLen) return Z_BUF_ERROR;

    stream.next_out = dest;
    stream.avail_out = (uInt)*destLen;
    if ((uLong)stream.avail_out != *destLen) return Z_BUF_ERROR;

    stream.zalloc = (alloc_func)0;
    stream.zfree = (free_func)0;

    err = inflateInit(&stream);
    if (err != Z_OK) return err;

    err = inflate(&stream, Z_FINISH);
    if (err != Z_STREAM_END) {
        inflateEnd(&stream);
        return err == Z_OK ? Z_BUF_ERROR : err;
    }
    *destLen = stream.total_out;

    err = inflateEnd(&stream);
    return err;
}
```

```
Microsoft Developer Studio Workspace File, Format Version 5.00
# WARNING: DO NOT EDIT OR DELETE THIS WORKSPACE FILE!

###############################################################################

Project: "ZLib"=.\ZLib.dsp - Package Owner=<4>

Package=<5>
{{{
}}}

Package=<4>
{{{
}}}

###############################################################################

Global:

Package=<5>
{{{
}}}

Package=<3>
{{{
}}}

###############################################################################
```

```
SOURCE=.\zlib.h
# End Source File
# Begin Source File

SOURCE=.\zutil.c
# End Source File
# Begin Source File

SOURCE=.\zutil.h
# End Source File
# End Target
# End Project
```

[illegible]

```
# Begin Source File

SOURCE=.\inffast.h
# End Source File
# Begin Source File

SOURCE=.\inffixed.h
# End Source File
# Begin Source File

SOURCE=.\inflate.c
# End Source File
# Begin Source File

SOURCE=.\inftrees.c
# End Source File
# Begin Source File

SOURCE=.\inftrees.h
# End Source File
# Begin Source File

SOURCE=.\infutil.c
# End Source File
# Begin Source File

SOURCE=.\infutil.h
# End Source File
# Begin Source File

SOURCE=.\trees.c
# End Source File
# Begin Source File

SOURCE=.\trees.h
# End Source File
# Begin Source File

SOURCE=.\uncompr.c
# End Source File
# Begin Source File

SOURCE=.\zconf.h
# End Source File
# Begin Source File
# Name "ZLib - Win32 Release"
# Name "ZLib - Win32 Debug"
# Begin Source File

SOURCE=.\adler32.c
# End Source File
# Begin Source File

SOURCE=.\compress.c
# End Source File
# Begin Source File

SOURCE=.\crc32.c
# End Source File
# Begin Source File

SOURCE=.\deflate.c
# End Source File
# Begin Source File

SOURCE=.\deflate.h
# End Source File
# Begin Source File

SOURCE=.\gzio.c
# End Source File
# Begin Source File

SOURCE=.\infblock.c
# End Source File
# Begin Source File

SOURCE=.\infblock.h
# End Source File
# Begin Source File

SOURCE=.\infcodes.c
# End Source File
# Begin Source File

SOURCE=.\infcodes.h
# End Source File
# Begin Source File

SOURCE=.\inffast.c
# End Source File
```

09529370 . 072400

```
# PROP Intermediate_Dir "Release"
# PROP Target_Dir ""
# ADD BASE CPP /nologo /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_WIN
DOWS" /YX /FD /c
# ADD CPP /nologo /Zp2 /MT /W3 /Ox /Os /D "WIN32" /D "NDEBUG" /D "
_WINDOWS" /FD /c
# SUBTRACT CPP /Gf /Gy /YX
# ADD BASE RSC /l 0x409
# ADD RSC /l 0x409
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LIB32=link.exe -lib
# ADD BASE LIB32 /nologo
# ADD LIB32 /nologo

!ELSEIF  "$(CFG)" == "ZLib - Win32 Debug"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "Debug"
# PROP BASE Intermediate_Dir "Debug"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Debug"
# PROP Intermediate_Dir "Debug"
# PROP Target_Dir ""
# ADD BASE CPP /nologo /W3 /GX /Z7 /Od /D "WIN32" /D "_DEBUG" /D "
_WINDOWS" /YX /FD /c
# ADD CPP /nologo /Zp2 /MTd /W3 /GX /Z7 /Od /D "WIN32" /D "_DEBUG"
 /D "_WINDOWS" /FD /c
# SUBTRACT CPP /YX
# ADD BASE RSC /l 0x409
# ADD RSC /l 0x409
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LIB32=link.exe -lib
# ADD BASE LIB32 /nologo
# ADD LIB32 /nologo

!ENDIF

# Begin Target
# Microsoft Developer Studio Project File - Name="ZLib" - Package
Owner=<4>
# Microsoft Developer Studio Generated Build File, Format Version
6.00
# ** DO NOT EDIT **

# TARGTYPE "Win32 (x86) Static Library" 0x0104

CFG=ZLib - Win32 Debug
!MESSAGE This is not a valid makefile. To build this project using
 NMAKE,
!MESSAGE use the Export Makefile command and run
!MESSAGE
!MESSAGE NMAKE /f "ZLib.mak".
!MESSAGE
!MESSAGE You can specify a configuration when running NMAKE
!MESSAGE by defining the macro CFG on the command line. For exampl
e:
!MESSAGE
!MESSAGE NMAKE /f "ZLib.mak" CFG="ZLib - Win32 Debug"
!MESSAGE
!MESSAGE Possible choices for configuration are:
!MESSAGE
!MESSAGE "ZLib - Win32 Release" (based on "Win32 (x86) Static Libr
ary")
!MESSAGE "ZLib - Win32 Debug" (based on "Win32 (x86) Static Librar
y")
!MESSAGE

# Begin Project
# PROP AllowPerConfigDependencies 0
# PROP Scc_ProjName ""$/ZLib", LBAAAAAA"
# PROP Scc_LocalPath "."
CPP=cl.exe
RSC=rc.exe

!IF  "$(CFG)" == "ZLib - Win32 Release"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "Release"
# PROP BASE Intermediate_Dir "Release"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Release"
```

```
{{{
    begin source code control
    "$/zcomp", RXAAAAAA
    .
    end source code control
}}}

Package=<3>
{{{
}}}

###################################################################
#############
```

```
Microsoft Developer Studio Workspace File, Format Version 5.00
# WARNING: DO NOT EDIT OR DELETE THIS WORKSPACE FILE!

##################################################################
#############

Project: "ZLib"=..\ZLib\ZLib.dsp - Package Owner=<4>

Package=<5>
{{{
    begin source code control
    "$/ZLib", LBAAAAAA
    ..\zlib
    end source code control
}}}

Package=<4>
{{{
}}}

##################################################################
#############

Project: "zcomp"=.\zcomp.dsp - Package Owner=<4>

Package=<5>
{{{
    begin source code control
    "$/zcomp", RXAAAAAA
    .
    end source code control
}}}

Package=<4>
{{{
    Begin Project Dependency
    Project_Dep_Name ZLib
    End Project Dependency
}}}

##################################################################
#############

Global:

Package=<5>
```

```
ine:I386 /pdbtype:sept
# ADD LINK32 /nologo /subsystem:console /debug /machine:I386 /pdbt
ype:sept

!ENDIF

# Begin Target

# Name "zcomp - Win32 Release"
# Name "zcomp - Win32 Debug"
# Begin Source File

SOURCE=.\zcomp.cpp
# End Source File
# End Target
# End Project
```

[illegible]

```
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_CON
SOLE" /D "_MBCS" /YX /FD /c
# ADD CPP /nologo /Zp2 /MT /W3 /GX /O2 /I "..\ZLib" /D "WIN32" /D
"NDEBUG" /D "_CONSOLE" /D "_MBCS" /FD /c
# SUBTRACT CPP /YX
# ADD BASE RSC /l 0x409 /d "NDEBUG"
# ADD RSC /l 0x409 /d "NDEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib c
omdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.l
ib odbc32.lib odbccp32.lib /nologo /subsystem:console /machine:I38
6
# ADD LINK32 /nologo /subsystem:console /machine:I386

!ELSEIF  "$(CFG)" == "zcomp - Win32 Debug"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "Debug"
# PROP BASE Intermediate_Dir "Debug"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Debug"
# PROP Intermediate_Dir "Debug"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /W3 /Gm /GX /Zi /Od /D "WIN32" /D "_DEBUG"
/D "_CONSOLE" /D "_MBCS" /YX /FD /c
# ADD CPP /nologo /Zp2 /MTd /W3 /Gm /GX /Zi /Od /I "..\ZLib" /D "W
IN32" /D "_DEBUG" /D "_CONSOLE" /D "_MBCS" /FD /c
# SUBTRACT CPP /YX
# ADD BASE RSC /l 0x409 /d "_DEBUG"
# ADD RSC /l 0x409 /d "_DEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib c
omdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.l
ib odbc32.lib odbccp32.lib /nologo /subsystem:console /debug /mach
```

```
# Microsoft Developer Studio Project File - Name="zcomp" - Package
 Owner=<4>
# Microsoft Developer Studio Generated Build File, Format Version
5.00
# ** DO NOT EDIT **

# TARGTYPE "Win32 (x86) Console Application" 0x0103

CFG=zcomp - Win32 Debug
!MESSAGE This is not a valid makefile. To build this project using
 NMAKE,
!MESSAGE use the Export Makefile command and run
!MESSAGE
!MESSAGE NMAKE /f "zcomp.mak".
!MESSAGE
!MESSAGE You can specify a configuration when running NMAKE
!MESSAGE by defining the macro CFG on the command line. For exampl
e:
!MESSAGE
!MESSAGE NMAKE /f "zcomp.mak" CFG="zcomp - Win32 Debug"
!MESSAGE
!MESSAGE Possible choices for configuration are:
!MESSAGE
!MESSAGE "zcomp - Win32 Release" (based on "Win32 (x86) Console Ap
plication")
!MESSAGE "zcomp - Win32 Debug" (based on "Win32 (x86) Console Appl
ication")
!MESSAGE

# Begin Project
# PROP Scc_ProjName ""$/zcomp", RXAAAAAA"
# PROP Scc_LocalPath "."
CPP=cl.exe
RSC=rc.exe

!IF  "$(CFG)" == "zcomp - Win32 Release"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "Release"
# PROP BASE Intermediate_Dir "Release"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Release"
# PROP Intermediate_Dir "Release"
```

```
        fread (pBytesIn, lFileLen, 1, input);
        fclose (input);

        fwrite (&lFileLen, sizeof (lFileLen), 1, output);

        Bytef *pBytesOut = (Bytef *) malloc (lFileLen * 2);

        uLong lCompressedLen;
        compress2 (pBytesOut, &lCompressedLen, pBytesIn, lFileLen,
          Z_BEST_COMPRESSION);
        fwrite (pBytesOut, lCompressedLen, 1, output);
        fclose (output);

        free (pBytesIn);
        free (pBytesOut);
    }

    exit (0);
}
```

[illegible]

```
#include <stdio.h>
#include "zlib.h"
#include <string.h>
#include <stdlib.h>


void main (int argc, char *argv[])
{
    if (argc < 3)
    {
        printf ("Usage: %s [-d] input output\n", argv[0]);
        exit (0);
    }

    if (strcmp (argv[1], "-d") == 0)
    {
        FILE *input;
        FILE *output;

        if ((input = fopen (argv[2], "rb")) == NULL)
        {
            printf ("Can't open file %s\n", argv[2]);
            exit (0);
        }

        if ((output = fopen (argv[3], "wb")) == NULL)
        {
            printf ("Can't open file %s\n", argv[3]);
            fclose (input);
            exit (0);
        }

        fseek (input, 0, SEEK_END);
        uLong lFileLen = ftell (input);
        fseek (input, 0, SEEK_SET);

        Bytef *pBytesIn = (Bytef *) malloc (lFileLen);
        fread (pBytesIn, lFileLen, 1, input);
        fclose (input);

        uLong lUncompressedLen = *((long *) pBytesIn);
        Bytef *pBytesOut = (Bytef *) malloc (lUncompressedLen);
        uncompress (pBytesOut, &lUncompressedLen, pBytesIn + 4, lFileLen - 4);

        fwrite (pBytesOut, lUncompressedLen, 1, output);
        fclose (output);

        free (pBytesIn);
        free (pBytesOut);
    }
    else
    {
        FILE *input;
        FILE *output;

        if ((input = fopen (argv[1], "rb")) == NULL)
        {
            printf ("Can't open file %s\n", argv[1]);
            exit (0);
        }

        if ((output = fopen (argv[2], "wb")) == NULL)
        {
            printf ("Can't open file %s\n", argv[2]);
            fclose (input);
            exit (0);
        }

        fseek (input, 0, SEEK_END);
        uLong lFileLen = ftell (input);
        fseek (input, 0, SEEK_SET);

        Bytef *pBytesIn = (Bytef *) malloc (lFileLen);
```

APPENDIX

Inventors: Bruce T. Petro, Andrew Cohen and Jason Sulak

Title: ON-LINE SYSTEM FOR CREATING A PRINTABLE PRODUCT

```
#ifndef __WAITDLG_H_
#define __WAITDLG_H_

#include "resource.h"

/////////////////////////////////////////////////////////////////////////////
// CWaitDlg
class CWaitDlg :
    public CDialogImpl<CWaitDlg>
{
public:
    CWaitDlg();
    ~CWaitDlg();

    enum { IDD = IDD_WAITDLG };

BEGIN_MSG_MAP(CWaitDlg)
    MESSAGE_HANDLER(WM_INITDIALOG, OnInitDialog)
    COMMAND_ID_HANDLER(IDCANCEL, OnCancel)
END_MSG_MAP()

    LRESULT OnCancel(WORD wNotifyCode, WORD wID, HWND hWndCtl, BOOL& bHandled);
    LRESULT OnInitDialog(UINT uMsg, WPARAM wParam, LPARAM lParam, BOOL& bHandled);
};

#endif //__WAITDLG_H_
```

09623370 .072400

```
//============================================================================//
//============================================================================//
#include "stdafx.h"
#include "WaitDlg.h"

//----------------------------------------------------------------------------//
//----------------------------------------------------------------------------//
CWaitDlg::CWaitDlg()
{
}

//----------------------------------------------------------------------------//
//----------------------------------------------------------------------------//
CWaitDlg::~CWaitDlg()
{
}

//----------------------------------------------------------------------------//
//----------------------------------------------------------------------------//
LRESULT CWaitDlg::OnCancel(WORD, WORD, HWND, BOOL &)
{
    return 0;
}

//----------------------------------------------------------------------------//
//----------------------------------------------------------------------------//
LRESULT CWaitDlg::OnInitDialog(UINT, WPARAM, LPARAM, BOOL &)
{
    CenterWindow ();
    return 1;
}
```

```
//
// Don't forget to update the version number in Ctp.rgs, AxCtp.inf, NpCtp.js
//

#define VER_COMPANY                "American Greetings.com\0"
#define VER_COPYRIGHT              "Copyright © 1999 American Greetings.com\0"

#define VER_PRODUCT_VERSION        1,0,0,0
#define VER_PRODUCT_VERSION_STR    "1.0"
#define VER_FILE_VERSION           1,0,0,0
#define VER_FILE_VERSION_STR       "1.0"
```

```
// stdafx.h : include file for standard system include files,
//      or project specific include files that are used frequently,
//      but are changed infrequently

#if !defined(AFX_STDAFX_H__4B81C688_1084_11D3_9330_00104BC4A611__INCLUDED_)
#define AFX_STDAFX_H__4B81C688_1084_11D3_9330_00104BC4A611__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

#define STRICT


#define _WIN32_WINNT 0x0400
#define _ATL_APARTMENT_THREADED


#include <atlbase.h>
//You may derive a class from CComModule and use it if you want to override
//something, but do not change the name of _Module
extern CComModule _Module;
#include <atlcom.h>
#include <atlctl.h>
#include <atlwin.h>

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately before the previous li
ne.

#include <vector>

#define WIDTH(r)          ((r).right - (r).left)
#define HEIGHT(r)         ((r).bottom - (r).top)
#define SWAP(a,b)         ((a) ^= (b),(b) ^= (a),(a) ^= (b))
#define APP_RESOLUTION    1440


#endif // !defined(AFX_STDAFX_H__4B81C688_1084_11D3_9330_00104BC4A611__INCLUDED)
```

```
// stdafx.cpp : source file that includes just the standard includes
//  stdafx.pch will be the pre-compiled header
//  stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"

#ifdef _ATL_STATIC_REGISTRY
#include <statreg.h>
#include <statreg.cpp>
#endif

#include <atlimpl.cpp>
#include <atlctl.cpp>
#include <atlwin.cpp>
```

```
//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by AxCtp.rc
//
#define IDS_PROJNAME                    100
#define IDR_CTP                         101
#define IDD_CTLPANEL                    102
#define IDD_WAITDLG                     103
#define IDC_PAGE1                       201
#define IDR_AGLOGO                      201
#define IDC_PAGE2                       202
#define IDR_CPLOGO                      202
#define IDC_PAGE3                       203
#define IDR_CACFC                       203
#define IDC_PAGE4                       204
#define IDD_DBLSIDEINTRO                204
#define IDC_FONT                        205
#define IDD_DBLSIDESTEP1                205
#define IDC_PTSIZE                      206
#define IDD_DBLSIDEEND                  206
#define IDC_COLOR                       207
#define IDD_DBLSIDESTEP2                207
#define IDC_LEFT                        208
#define IDB_1UP                         208
#define IDC_CENTER                      209
#define IDB_2UP                         209
#define IDC_RIGHT                       210
#define IDB_3UP                         210
#define IDC_PRINT                       211
#define IDB_2DOWN                       211
#define IDC_SINGLEFOLD                  212
#define IDB_1UP2DOWN                    212
#define IDC_QUARTERFOLD                 213
#define IDC_FRAME1                      214
#define IDC_FRAME2                      215
#define IDC_FRAME3                      216
#define IDC_FRAME4                      217
#define IDC_DBLSIDE                     218

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        213
#define _APS_NEXT_COMMAND_VALUE         32768
#define _APS_NEXT_CONTROL_VALUE         219
#define _APS_NEXT_SYMED_VALUE           104
#endif
#endif
```

```
#ifndef __PROP_SHEET_H__
#define __PROP_SHEET_H__

#include <commctrl.h>

template <class T>
class ATL_NO_VTABLE CPropertyPageImpl : public CDialogImplBase
{
public:
    PROPSHEETPAGE m_psp;

    operator PROPSHEETPAGE*() { return &m_psp; }

// Construction
    CPropertyPageImpl(LPCTSTR lpszTitle = NULL)
    {
        // initialize PROPSHEETPAGE struct
        memset(&m_psp, 0, sizeof(PROPSHEETPAGE));
        m_psp.dwSize = sizeof(PROPSHEETPAGE);
        m_psp.dwFlags = PSP_USECALLBACK;
        m_psp.hInstance = _Module.GetResourceInstance();
        m_psp.pszTemplate = MAKEINTRESOURCE(T::IDD);
        m_psp.pfnDlgProc = (DLGPROC)T::StartDialogProc;
        m_psp.pfnCallback = T::PropPageCallback;
        m_psp.lParam = (LPARAM)this;

        if(lpszTitle != NULL)
        {
            m_psp.pszTitle = lpszTitle;
            m_psp.dwFlags |= PSP_USETITLE;
        }
    }

    static UINT CALLBACK PropPageCallback(HWND hWnd, UINT uMsg, LPPROPSHEETPAGE ppsp)
    {
        _ASSERTE(hWnd == NULL);
        if(uMsg == PSPCB_CREATE)
        {
            CDialogImplBase* pPage = (CDialogImplBase*)ppsp->lParam;
            _Module.AddCreateWndData(&pPage->m_thunk.cd, pPage);
        }
        return 1;
    }

    HPROPSHEETPAGE Create()
    {
        return ::CreatePropertySheetPage(&m_psp);
    }

    BOOL EndDialog(int)
    {
        // do nothing here, calling ::EndDialog will close the whole sheet
        _ASSERTE(FALSE);
        return FALSE;
    }

// Operations
    void CancelToClose()
    {
        _ASSERTE(::IsWindow(m_hWnd));
        _ASSERTE(GetParent() != NULL);

        ::SendMessage(GetParent(), PSM_CANCELTOCLOSE, 0, 0L);
    }
    void SetModified(BOOL bChanged = TRUE)
    {
        _ASSERTE(::IsWindow(m_hWnd));
        _ASSERTE(GetParent() != NULL);

        if(bChanged)
            ::SendMessage(GetParent(), PSM_CHANGED, (WPARAM)m_hWnd, 0L);
        else
            ::SendMessage(GetParent(), PSM_UNCHANGED, (WPARAM)m_hWnd, 0L);
    }
    void SetWizardButtons (DWORD dwFlags)
    {
        _ASSERTE(::IsWindow(m_hWnd));
        _ASSERTE(GetParent() != NULL);

        ::PostMessage(GetParent(), PSM_SETWIZBUTTONS, 0, (LPARAM) dwFlags);
    }
    LRESULT QuerySiblings(WPARAM wParam, LPARAM lParam)
    {
        _ASSERTE(::IsWindow(m_hWnd));
        _ASSERTE(GetParent() != NULL);

        return ::SendMessage(GetParent(), PSM_QUERYSIBLINGS, wParam, lParam);
    }

    BEGIN_MSG_MAP(CPropertyPageImpl<T>)
        MESSAGE_HANDLER(WM_NOTIFY, OnNotify)
    END_MSG_MAP()

// Message handler
    LRESULT OnNotify(UINT uMsg, WPARAM wParam, LPARAM lParam, BOOL& bHandled)
    {
        _ASSERTE(::IsWindow(m_hWnd));
        NMHDR* pNMHDR = (NMHDR*)lParam;

        // don't handle messages not from the page/sheet itself
        if(pNMHDR->hwndFrom != m_hWnd && pNMHDR->hwndFrom != ::GetParent(m_hWnd))
        {
            bHandled = FALSE;
            return 1;
        }

        T* pT = (T*)this;
        LRESULT lResult = 0;
        // handle default
        switch(pNMHDR->code)
        {
        case PSN_SETACTIVE:
            lResult = pT->OnSetActive() ? 0 : -1;
            break;
        case PSN_KILLACTIVE:
            lResult = !pT->OnKillActive();
            break;
        case PSN_APPLY:
            lResult = pT->OnApply() ? PSNRET_NOERROR : PSNRET_INVALID_NOCHANGEPAGE;
            break;
        case PSN_RESET:
            pT->OnReset();
            break;
        case PSN_QUERYCANCEL:
            lResult = !pT->OnQueryCancel();
            break;
        case PSN_WIZNEXT:
            lResult = !pT->OnWizardNext();
            break;
        case PSN_WIZBACK:
            lResult = !pT->OnWizardBack();
            break;
        case PSN_WIZFINISH:
            lResult = !pT->OnWizardFinish();
            break;
        case PSN_HELP:
            lResult = pT->OnHelp();
            break;
        default:
            bHandled = FALSE;   // not handled
        }

        return lResult;
    }

// Overridables
```

```
    BOOL OnSetActive()
    {
        return TRUE;
    }
    BOOL OnKillActive()
    {
        return TRUE;
    }
    BOOL OnApply()
    {
        return TRUE;
    }
    void OnReset()
    {
    }
    BOOL OnQueryCancel()
    {
        return TRUE;    // ok to cancel
    }
    BOOL OnWizardBack()
    {
        return TRUE;
    }
    BOOL OnWizardNext()
    {
        return TRUE;
    }
    BOOL OnWizardFinish()
    {
        return TRUE;
    }
    BOOL OnHelp()
    {
        return TRUE;
    }
};

#endif //__PROP_SHEET_H__
```

```
#ifndef __FONT_H_
#define __FONT_H_

#include "AGDoc.h"

typedef struct
{
    LOGFONT lf;
    char    szFullName[LF_FULLFACESIZE];
} FONTINFO;
typedef std::vector<FONTINFO> FONTARRAY;

typedef struct
{
    char    szFontFile[_MAX_FNAME];
    char    szFullName[LF_FULLFACESIZE];
    BYTE    *pDownloadData;
    DWORD   dwDownloadSize;
} FONTDOWNLOAD;
typedef std::vector<FONTDOWNLOAD> FONTDOWNLOADARRAY;


class CFontList
{
public:
    CFontList ();

    void CheckFonts (LOGFONTARRAY &lfArray, FONTDOWNLOADARRAY &DownloadArray);
    HDC GetEnumFontDC ()
        { return (m_EnumFontDC); }
    FONTARRAY &GetFontArray ()
        { return (m_FontArray); }
    void InitFontArray ();
    void InstallFont (BYTE *pBytes, DWORD dwLen, const char *pszTypeFace,
      const char *pszTTFName);

protected:
    void CheckDefaultFont ();

protected:
    FONTARRAY    m_FontArray;
    HDC          m_EnumFontDC;
};

#endif //__FONT_H_
```

```
        FONTINFO fi;
        fi.lf = pelf->elfLogFont;
        lstrcpy (fi.szFullName, (const char *) pelf->elfFullName);
        pFontList->GetFontArray ().push_back (fi);
    }

    return (1);
}

//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
static int CALLBACK EnumFontFamilyProc (ENUMLOGFONT *pelf,
  NEWTEXTMETRIC * /*pntm*/, int FontType, LPARAM lParam)
{
    if (FontType & TRUETYPE_FONTTYPE)
    {
        CFontList *pFontList = (CFontList *) lParam;
        EnumFontFamilies (pFontList->GetEnumFontDC (),
          pelf->elfLogFont.lfFaceName, (FONTENUMPROC) EnumFontFamilyProc2,
          lParam);
    }

    return (1);
}

//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
CFontList::CFontList ()
{
    m_EnumFontDC = NULL;
    InitFontArray ();
    CheckDefaultFont ();
}

//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
void CFontList::CheckDefaultFont ()
{
    int nFonts = m_FontArray.size ();
    for (int i = 0; i < nFonts; i++)
    {
        if (lstrcmp (m_FontArray[i].lf.lfFaceName, DEFAULT_FONT) == 0)
            break;
    }

    if (i >= nFonts)
    {
        HGLOBAL hResFont = NULL;
        HRSRC   hResource;

        if ((hResource = ::FindResource (_Module.GetResourceInstance (),
          MAKEINTRESOURCE (IDR_CACFC), "TTZ")) != NULL)
        {
            hResFont = ::LoadResource (_Module.GetResourceInstance (),
              hResource);
        }

        if (hResFont)
        {
            BYTE *pCompressedBytes = (BYTE *) ::LockResource (hResFont);
            DWORD dwCompressedLen = ::SizeofResource (_Module.GetResourceInstance (),
              hResource);
            InstallFont (pCompressedBytes, dwCompressedLen, DEFAULT_FONT,
              DEFAULT_FONTFILE);
        }
    }
}

//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
void CFontList::CheckFonts (LOGFONTARRAY &lfArray,
  FONTDOWNLOADARRAY &DownloadArray)
{
    int nCheckFonts = lfArray.size ();
    for (int i = 0; i < nCheckFonts; i++)
    {
        int nFonts = m_FontArray.size ();
        for (int j = 0; j < nFonts; j++)
        {
            if (lstrcmp (lfArray[i].lfFaceName, m_FontArray[j].lf.lfFaceName) == 0 &&
              lfArray[i].lfWeight == m_FontArray[j].lf.lfWeight &&
              (lfArray[i].lfItalic != 0) == (m_FontArray[j].lf.lfItalic != 0))
            {
                break;
            }
        }

        if (j >= nFonts)
        {
            char szFont[LF_FULLFACESIZE];
            lstrcpy (szFont, lfArray[i].lfFaceName);
            if (lfArray[i].lfWeight == 700)
                lstrcat (szFont, " (Bold)");
            if (lfArray[i].lfItalic != 0)
                lstrcat (szFont, " (Italic)");

            for (int k = 0; FontNames[k].pszFont; k++)
            {
                if (lstrcmp (FontNames[k].pszFont, szFont) == 0)
                {
                    FONTDOWNLOAD fd;
                    lstrcpy (fd.szFontFile, FontNames[k].pszFontFile);
                    lstrcpy (fd.szFullName, FontNames[k].pszFullName);
                    fd.pDownloadData = NULL;
                    fd.dwDownloadSize = 0;
                    DownloadArray.push_back (fd);
                    break;
                }
            }
        }
    }

//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CFontList::InitFontArray ()
{
    m_FontArray.clear ();

    m_EnumFontDC = ::CreateIC ("DISPLAY", NULL, NULL, NULL);
    EnumFontFamilies (m_EnumFontDC, NULL, (FONTENUMPROC) EnumFontFamilyProc,
      (LPARAM) this);
    ::DeleteDC (m_EnumFontDC);
    m_EnumFontDC = NULL;
}

//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CFontList::InstallFont (BYTE *pBytes, DWORD dwLen, const char *pszTypeFace,
  const char *pszTTFName)
{
    bool bInstalled = false;

    DWORD dwUncompressedLen = *((DWORD *) pBytes);
    BYTE *pUncompressedBytes = (BYTE *) malloc (dwUncompressedLen);
    uncompress (pUncompressedBytes, &dwUncompressedLen, pBytes + 4, dwLen - 4);

    char szTTFName[15];
    char szTTFDir[_MAX_PATH];
    char szTTFPath[_MAX_PATH];
    ::GetWindowsDirectory (szTTFDir, sizeof (szTTFDir));
    if (szTTFDir[lstrlen (szTTFDir) - 1] != '\\')
        lstrcat (szTTFDir, "\\");

    lstrcpy (szTTFName, pszTTFName);
    szTTFName[lstrlen (szTTFName) - 1] = 'F';
    OSVERSIONINFO osvi;
    osvi.dwOSVersionInfoSize = sizeof (OSVERSIONINFO);
    ::GetVersionEx (&osvi);
    if (osvi.dwPlatformId == VER_PLATFORM_WIN32_WINDOWS ||
      (osvi.dwPlatformId == VER_PLATFORM_WIN32_NT && osvi.dwMajorVersion >= 4))
    {
        lstrcat (szTTFDir, "Fonts\\");
    }
    else
    {
        lstrcat (szTTFDir, "System\\");
    }

    HANDLE  hf;
    int     nCount = 0;
    do
    {
        if (nCount > 0)
            szTTFName[7] = (char) ('0' + nCount);
        lstrcpy (szTTFPath, szTTFDir);
        lstrcat (szTTFPath, szTTFName);
        hf = ::CreateFile (szTTFPath, GENERIC_WRITE, 0, NULL,
          CREATE_NEW, FILE_ATTRIBUTE_NORMAL, NULL);
    } while (hf == INVALID_HANDLE_VALUE && ++nCount < 10);

    if (hf != INVALID_HANDLE_VALUE)
    {
        DWORD dwWritten;
        ::WriteFile (hf, pUncompressedBytes, dwUncompressedLen, &dwWritten, NULL);
        ::CloseHandle (hf);

        if (::AddFontResource (szTTFPath))
        {
            LPCTSTR lpSubKey;
            if (osvi.dwPlatformId == VER_PLATFORM_WIN32_WINDOWS)
                lpSubKey = FONT_SUBKEY;
            else
                lpSubKey = NTFONT_SUBKEY;

            HKEY hKey;
            if (::RegOpenKeyEx (HKEY_LOCAL_MACHINE, lpSubKey,
              0, KEY_WRITE, &hKey) == ERROR_SUCCESS)
            {
                char szFaceName[100];
                lstrcpy (szFaceName, pszTypeFace);
                lstrcat (szFaceName, " (TrueType)");

                ::RegSetValueEx (hKey, szFaceName, 0, REG_SZ,
                  (BYTE *) szTTFName, lstrlen (szTTFName) + 1);
                ::RegCloseKey(hKey);

                bInstalled = true;
            }

            ::SendMessage (HWND_BROADCAST, WM_FONTCHANGE, 0, 0);
        }
    }
    free (pUncompressedBytes);

    if (! bInstalled)
    {
        char szMsg[128];
        wsprintf (szMsg, "Unable to install font '%s' on your system.",
          pszTypeFace);
        ::MessageBox (NULL, szMsg, "Font Install", MB_OK);
    }
}
```

```
//=========================================================================//
//=========================================================================//
#include "stdafx.h"
#include "font.h"
#include "resource.h"
#include "zutil.h"


#ifdef _AFX
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
#endif

#define DEFAULT_FONT        "CAC Futura Casual"
#define DEFAULT_FONTFILE    "CACFC___.TTF"
#define FONT_SUBKEY         "Software\\Microsoft\\Windows\\CurrentVersion\\Fonts"
#define NTFONT_SUBKEY       "Software\\Microsoft\\Windows NT\\CurrentVersion\\Fonts"

typedef struct
{
    const char  *pszFont;
    const char  *pszFullName;
    const char  *pszFontFile;
} FONTNAMES;


const static FONTNAMES FontNames[] =
{
    "CAC Camelot",                  "CAC Camelot",                  "CACCAMEL.TTZ",
    "CAC Champagne",                "CAC Champagne",                "CACCHAMP.TTZ",
    "CAC Futura Casual",            "CAC Futura Casual",            "CACFC___.TTZ",
    "CAC Futura Casual Bold",       "CAC Futura Casual Bold",       "CACFCB__.TTZ",
    "CAC Futura Casual Bold Italic","CAC Futura Casual Bold Italic","CACFCBI_.TTZ",
    "CAC Futura Casual Med. Italic","CAC Futura Casual Med. Italic","CACFCMI_.TTZ",
    "CAC Krazy Legs",               "CAC Krazy Legs",               "CACKL___.TTZ",
    "CAC Krazy Legs Bold",          "CAC Krazy Legs Bold",          "CACKLB__.TTZ",
    "CAC Lasko Condensed",          "CAC Lasko Condensed",          "CACLC___.TTZ",
    "CAC Lasko Even Weight",        "CAC Lasko Even Weight",        "CACLEW__.TTZ",
    "CAC Leslie",                   "CAC Leslie",                   "CACLESLI.TTZ",
    "CAC Logo Alternate",           "CAC Logo Alternate",           "CACLA___.TTZ",
    "CAC Moose",                    "CAC Moose",                    "CACMOOSE.TTZ",
    "CAC Norm Heavy",               "CAC Norm Heavy",               "CACNH___.TTZ",
    "CAC One Seventy",              "CAC One Seventy",              "CACOS___.TTZ",
    "CAC Pinafore",                 "CAC Pinafore",                 "CACPINAF.TTZ",
    "CAC Saxon Bold",               "CAC Saxon Bold",               "CACSB___.TTZ",
    "CAC Shishoni Brush",           "CAC Shishoni Brush",           "CACSHISH.TTZ",
    "CAC Valiant",                  "CAC Valiant",                  "CACVALIA.TTZ",
    "Cataneo BT",                   "Cataneo BT",                   "TT0952M_.TTZ",
    "DomCasual BT",                 "Dom Casual BT",                "TT0604M_.TTZ",
    "Freehand575 BT",               "Freehand 575 BT",              "TT1046M_.TTZ",
    "GoudyOlSt BT (Italic)",        "Goudy Old Style Italic BT",    "TT0108M_.TTZ",
    "Informal011 BT",               "Informal 011 BT",              "TT1115M_.TTZ",
    "Lydian Csv BT",                "Lydian Cursive BT",            "TT0845M_.TTZ",
    "MattAntique BT",               "Matt Antique BT",              "TT1014M_.TTZ",
    "MattAntique BT (Italic)",      "Matt Antique Italic BT",       "TT1015M_.TTZ",
    "Snell BT",                     "Snell BT",                     "TT0196M_.TTZ",
    "Sprocket BT",                  "Sprocket BT",                  "TT1244M_.TTZ",
    "Swis721 BT (Bold)",            "Swiss 721 Bold BT",            "TT0005M_.TTZ",
    "Swis721 Md BT",                "Swiss 721 Medium BT",          "TT0759M_.TTZ",
    NULL,                           NULL,                           NULL
};

//-------------------------------------------------------------------------//
//-------------------------------------------------------------------------//
static int CALLBACK EnumFontFamilyProc2 (ENUMLOGFONT *pelf,
  NEWTEXTMETRIC * /*pntm*/, int FontType, LPARAM lParam)
{
    if (FontType & TRUETYPE_FONTTYPE)
    {
        CFontList *pFontList = (CFontList *) lParam;
```

```
/*********************************************************
   DllData file -- generated by MIDL compiler

        DO NOT ALTER THIS FILE

   This file is regenerated by MIDL on every IDL file compile.

   To completely reconstruct this file, delete it and rerun MIDL
   on all the IDL files in this DLL, specifying this file for the
   /dlldata command line option

*********************************************************/

#define PROXY_DELEGATION

#include <rpcproxy.h>

#ifdef __cplusplus
extern "C"   {
#endif

EXTERN_PROXY_FILE( AxCtp )


PROXYFILE_LIST_START
/* Start of list */
  REFERENCE_PROXY_FILE( AxCtp ),
/* End of list */
PROXYFILE_LIST_END


DLLDATA_ROUTINES( aProxyFileList, GET_DLL_CLSID )

#ifdef __cplusplus
}  /*extern "C" */
#endif

/* end of generated dlldata file */
```

```
#ifndef __DBLSIDE_H_
#define __DBLSIDE_H_

#include "propsht.h"
#include "resource.h"

class CSelectFrame : public CWindowImpl<CSelectFrame>
{
public:

BEGIN_MSG_MAP (CSelectFrame)
    MESSAGE_HANDLER (WM_LBUTTONDOWN, OnLButtonDown)
    MESSAGE_HANDLER (WM_NCHITTEST, OnNCHitTest)
END_MSG_MAP ()

public:
    LRESULT OnLButtonDown (UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM /*lParam*/, BOOL & bHandled)
    {
        ::SendMessage (GetParent (), WM_COMMAND,
          MAKEWPARAM (GetDlgCtrlID (), BN_CLICKED), (LPARAM) m_hWnd);
        bHandled = TRUE;
        return (TRUE);
    }

    LRESULT OnNCHitTest (UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM /*lParam*/, BOOL & bHandled)
    {
        bHandled = TRUE;
        return (HTCLIENT);
    }
};

/////////////////////////////////////////////////////////////////////////////
// CDblSideIntro
class CDblSideIntro :
    public CPropertyPageImpl<CDblSideIntro>
{
public:
    enum { IDD = IDD_DBLSIDEINTRO };

    BOOL OnSetActive ();
    BOOL OnWizardNext ();

public:
    char    *m_pszDriver;
    char    *m_pszDevice;
    char    *m_pszOutput;
    DEVMODE *m_pDevMode;
};


/////////////////////////////////////////////////////////////////////////////
// CDblSideStep1
class CDblSideStep1 :
    public CPropertyPageImpl<CDblSideStep1>
{
public:
    enum { IDD = IDD_DBLSIDESTEP1 };

    BOOL OnSetActive ();
    BOOL OnWizardNext ();

public:
    char    *m_pszDriver;
    char    *m_pszDevice;
    char    *m_pszOutput;
    DEVMODE *m_pDevMode;
};

/////////////////////////////////////////////////////////////////////////////
// CDblSideStep2
class CDblSideStep2 :
    public CPropertyPageImpl<CDblSideStep2>
{
```

```
public:
    enum { IDD = IDD_DBLSIDESTEP2 };

    CDblSideStep2 ()
        { m_nSelected = -1; }

    BEGIN_MSG_MAP (CDblSideStep2)
        COMMAND_HANDLER (IDC_FRAME1, BN_CLICKED, OnFrame)
        COMMAND_HANDLER (IDC_FRAME2, BN_CLICKED, OnFrame)
        COMMAND_HANDLER (IDC_FRAME3, BN_CLICKED, OnFrame)
        COMMAND_HANDLER (IDC_FRAME4, BN_CLICKED, OnFrame)
        CHAIN_MSG_MAP(CPropertyPageImpl<CDblSideStep2>)
    END_MSG_MAP()

    int GetSelected ()
        { return (m_nSelected); }
    LRESULT OnFrame (WORD wNotifyCode, WORD wID, HWND hWndCtl, BOOL& bHandled);
    BOOL OnKillActive ();
    BOOL OnSetActive ();
    BOOL OnWizardNext ();

protected:
    int             m_nSelected;
    CSelectFrame    m_Frame1;
    CSelectFrame    m_Frame2;
    CSelectFrame    m_Frame3;
    CSelectFrame    m_Frame4;
};

///////////////////////////////////////////////////////////////////////////////
// CDblSideEnd
class CDblSideEnd :
    public CPropertyPageImpl<CDblSideEnd>
{
public:
    enum { IDD = IDD_DBLSIDEEND };

    CDblSideEnd ()
        { m_bFinished = false; }

    bool IsFinished ()
        { return (m_bFinished); }
    BOOL OnSetActive ();
    BOOL OnWizardFinish ();

protected:
    bool m_bFinished;
};

#endif //__DBLSIDE_H_
```

```
//==============================================================================//
//==============================================================================//
#include "stdafx.h"
#include "DblSide.h"
#include "AGDC.h"
#include "AGSym.h"


//------------------------------------------------------------------------------//
//------------------------------------------------------------------------------//
static void DrawArrow(CAGDC *pDC, POINT Pt)
{
    POINT Pts[7];
    Pts[0] = Pt;
    Pts[1].x = Pts[0].x - (APP_RESOLUTION / 2);
    Pts[1].y = Pts[0].y + (APP_RESOLUTION / 2);
    Pts[2].x = Pts[1].x + (APP_RESOLUTION / 4);
    Pts[2].y = Pts[1].y;
    Pts[3].x = Pts[2].x;
    Pts[3].y = Pts[2].y + (APP_RESOLUTION / 2);
    Pts[4].x = Pts[3].x + (APP_RESOLUTION / 2);
    Pts[4].y = Pts[3].y;
    Pts[5].x = Pts[4].x;
    Pts[5].y = Pts[4].y - (APP_RESOLUTION / 2);
    Pts[6].x = Pts[5].x + (APP_RESOLUTION / 4);
    Pts[6].y = Pts[5].y;

    HBRUSH hOldBrush = (HBRUSH)::SelectObject(pDC->GetHDC(),
                                              ::GetStockObject(NULL_BRUSH));
    HPEN hPen = ::CreatePen(PS_SOLID, pDC->GetDeviceInfo ().m_nLogPixelsX / 50,
                            RGB(0, 0, 0));
    HPEN hOldPen = (HPEN) ::SelectObject(pDC->GetHDC(), hPen);

    pDC->Polygon(Pts, 7);

    ::SelectObject(pDC->GetHDC(), hOldBrush);
    ::SelectObject(pDC->GetHDC(), hOldPen);
    ::DeleteObject(hPen);
}


//------------------------------------------------------------------------------//
//------------------------------------------------------------------------------//
static void PrintTest(bool bFirstTest, char *pszDriver, char *pszDevice,
                      char *pszOutput, DEVMODE *pDevMode)
{
    short nSaveCopies = -1;

    if (pDevMode)
    {
        pDevMode->dmOrientation = DMORIENT_PORTRAIT;
        pDevMode->dmFields |= DM_ORIENTATION;

        if (pDevMode->dmCopies > 1)
        {
            nSaveCopies = pDevMode->dmCopies;
            pDevMode->dmCopies = 1;
        }
    }

    CAGDC *pDC = new CAGDC(pszDriver, pszDevice, pszOutput, pDevMode);
    CAGDCInfo di = pDC->GetDeviceInfo();

    if (pDevMode && pDevMode->dmOrientation == DMORIENT_PORTRAIT &&
        di.m_nHorzSize > di.m_nVertSize)
    {
        delete pDC;
        pDevMode->dmOrientation = DMORIENT_LANDSCAPE;
        pDC = new CAGDC(pszDriver, pszDevice, pszOutput, pDevMode);
        di = pDC->GetDeviceInfo();
    }
    else if (pDevMode && pDevMode->dmOrientation == DMORIENT_LANDSCAPE &&
        di.m_nVertSize > di.m_nHorzSize)
    {
        delete pDC;
        pDevMode->dmOrientation = DMORIENT_PORTRAIT;
        pDC = new CAGDC(pszDriver, pszDevice, pszOutput, pDevMode);
        di = pDC->GetDeviceInfo();
    }

    if (! pDC->StartDoc("Create and Print"))
    {
        delete pDC;
        if (pDevMode && nSaveCopies != -1)
            pDevMode->dmCopies = nSaveCopies;
        return;
    }

    if (! pDC->StartPage())
    {
        pDC->AbortDoc();
        delete pDC;
        if (pDevMode && nSaveCopies != -1)
            pDevMode->dmCopies = nSaveCopies;
        return;
    }

    if (bFirstTest)
    {
        POINT Pt = { di.m_PhysPageSize.cx / 2, di.m_nLogPixelsY };
        pDC->DPAtoVPA(&Pt, 1);
        DrawArrow(pDC, Pt);

        char szMsg[] = "Please put this page back into the printer\n\n"
            "with the printed side UP\n\n"
            "and the arrow pointing TOWARD the printer.";

        int nSpecOffset = 0;
        RECT MsgRect = { 0, di.m_nLogPixelsY * 3, di.m_PhysPageSize.cx,
          di.m_nLogPixelsY * 6 };
        pDC->DPAtoVPA(&MsgRect);

        LOGFONT MsgFont;
        memset(&MsgFont, 0, sizeof(MsgFont));
        lstrcpy(MsgFont.lfFaceName, "Arial");
        MsgFont.lfHeight = -(14 * APP_RESOLUTION / 72);
        CAGSpec MsgSpec(MsgFont, RGB(0, 0, 0), eRagCentered);

        CAGSymText *pAGSymText = new CAGSymText();
        pAGSymText->Create(MsgRect);
        pAGSymText->SetVertJust(eVertTop);
        pAGSymText->SetText(szMsg, lstrlen(szMsg), 1, &MsgSpec, &nSpecOffset);
        pAGSymText->Draw(*pDC);
        delete pAGSymText;
    }
    else
    {
        POINT Pt = { di.m_PhysPageSize.cx / 4, di.m_nLogPixelsY };
        pDC->DPAtoVPA(&Pt, 1);
        DrawArrow(pDC, Pt);

        Pt.x = (di.m_PhysPageSize.cx / 4) * 3;
        Pt.y = di.m_nLogPixelsY;
        pDC->DPAtoVPA(&Pt, 1);
        DrawArrow(pDC, Pt);
    }

    if (! pDC->EndPage())
    {
        pDC->AbortDoc();
        delete pDC;
        if (pDevMode && nSaveCopies != -1)
            pDevMode->dmCopies = nSaveCopies;
        return;
    }
    if (! pDC->EndDoc())
    {
        pDC->AbortDoc();
        delete pDC;
        if (pDevMode && nSaveCopies != -1)
            pDevMode->dmCopies = nSaveCopies;
        return;
    }

    delete pDC;

    if (pDevMode && nSaveCopies != -1)
        pDevMode->dmCopies = nSaveCopies;
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
BOOL CDblSideIntro::OnSetActive()
{
    CWindow Parent(GetParent());
    Parent.CenterWindow();

    SetWizardButtons(PSWIZB_NEXT);
    return (TRUE);
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
BOOL CDblSideIntro::OnWizardNext()
{
    PrintTest(true, m_pszDriver, m_pszDevice, m_pszOutput, m_pDevMode);
    return (TRUE);
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
BOOL CDblSideStep1::OnSetActive()
{
    SetWizardButtons(PSWIZB_BACK | PSWIZB_NEXT);
    return (TRUE);
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
BOOL CDblSideStep1::OnWizardNext()
{
    PrintTest(false, m_pszDriver, m_pszDevice, m_pszOutput, m_pDevMode);
    return (TRUE);
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
LRESULT CDblSideStep2::OnFrame(WORD /*wNotifyCode*/, WORD wID, HWND hWndCtl,
                               BOOL &bHandled)
{
    long lStyle;

    if (m_nSelected != -1)
    {
        HWND hWnd = GetDlgItem(m_nSelected + IDC_FRAME1 - 1);
        lStyle = ::GetWindowLong(hWnd, GWL_STYLE);
        lStyle &= ~SS_BLACKFRAME;
        lStyle |= SS_ETCHEDFRAME;
        ::SetWindowLong(hWnd, GWL_STYLE, lStyle);

        RECT r;
        ::GetClientRect(hWnd, &r);
        POINT Pts[2];
        Pts[0].x = r.left;
        Pts[0].y = r.top;
        Pts[1].x = r.right;
        Pts[1].y = r.bottom;
        ::MapWindowPoints(hWnd, m_hWnd, Pts, 2);
        r.left = Pts[0].x;
        r.top = Pts[0].y;
        r.right = Pts[1].x;
        r.bottom = Pts[1].y;
        InvalidateRect(&r);
    }
    else
        SetWizardButtons(PSWIZB_BACK | PSWIZB_NEXT);

    lStyle = ::GetWindowLong(hWndCtl, GWL_STYLE);
    lStyle &= ~SS_ETCHEDFRAME;
    lStyle |= SS_BLACKFRAME;
    ::SetWindowLong(hWndCtl, GWL_STYLE, lStyle);

    RECT r;
    ::GetClientRect(hWndCtl, &r);
    POINT Pts[2];
    Pts[0].x = r.left;
    Pts[0].y = r.top;
    Pts[1].x = r.right;
    Pts[1].y = r.bottom;
    ::MapWindowPoints(hWndCtl, m_hWnd, Pts, 2);
    r.left = Pts[0].x;
    r.top = Pts[0].y;
    r.right = Pts[1].x;
    r.bottom = Pts[1].y;
    InvalidateRect(&r);

    m_nSelected = wID - IDC_FRAME1 + 1;

    bHandled = TRUE;
    return (TRUE);
}


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
BOOL CDblSideStep2::OnKillActive()
{
    m_Frame1.UnsubclassWindow();
    m_Frame2.UnsubclassWindow();
    m_Frame3.UnsubclassWindow();
    m_Frame4.UnsubclassWindow();
    return (TRUE);
}


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
BOOL CDblSideStep2::OnSetActive()
{
    m_Frame1.SubclassWindow(GetDlgItem(IDC_FRAME1));
    m_Frame2.SubclassWindow(GetDlgItem(IDC_FRAME2));
    m_Frame3.SubclassWindow(GetDlgItem(IDC_FRAME3));
    m_Frame4.SubclassWindow(GetDlgItem(IDC_FRAME4));

    if (m_nSelected == -1)
        SetWizardButtons(PSWIZB_BACK);
    else
        SetWizardButtons(PSWIZB_BACK | PSWIZB_NEXT);

    return (TRUE);
}


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
BOOL CDblSideStep2::OnWizardNext()
```

```
{
    return (TRUE);
}


//-------------------------------------------------------------------------//
//-------------------------------------------------------------------------//
BOOL CDblSideEnd::OnSetActive()
{
    SetWizardButtons(PSWIZB_BACK | PSWIZB_FINISH);
    return (TRUE);
}


//-------------------------------------------------------------------------//
//-------------------------------------------------------------------------//
BOOL CDblSideEnd::OnWizardFinish()
{
    m_bFinished = true;
    return (TRUE);
}
```

[illegible]

```
HKCR
{
    Ctp.Ctp.1 = s 'Ctp Class'
    {
        CLSID = s '{38578BF0-0ABB-11D3-9330-0080C6F796A1}'
    }
    Ctp.Ctp = s 'Ctp Class'
    {
        CLSID = s '{38578BF0-0ABB-11D3-9330-0080C6F796A1}'
        CurVer = s 'Ctp.Ctp.1'
    }
    NoRemove CLSID
    {
        ForceRemove {38578BF0-0ABB-11D3-9330-0080C6F796A1} = s 'Ctp Class'
        {
            ProgID = s 'Ctp.Ctp.1'
            VersionIndependentProgID = s 'Ctp.Ctp'
            ForceRemove 'Programmable'
            InprocServer32 = s '%MODULE%'
            {
                val ThreadingModel = s 'Apartment'
            }
            ForceRemove 'Control'
            ForceRemove 'Programmable'
            ForceRemove 'Insertable'
            ForceRemove 'ToolboxBitmap32' = s '%MODULE%, 1'
            'MiscStatus' = s '0'
            {
                '1' = s '131473'
            }
            'TypeLib' = s '{38578BF1-0ABB-11D3-9330-0080C6F796A1}'
            'Version' = s '1.0'
        }
    }
```

```
#ifndef __CTP_H_
#define __CTP_H_

#include "AxCtp.h"
#include "resource.h"
#include "CtlPanel.h"
#include "AGDoc.h"
#include "AGSym.h"
#include "AGDC.h"
#include "Bsc2.h"
#include "Font.h"


#define dwSAFETY_OPTIONS    INTERFACESAFE_FOR_UNTRUSTED_CALLER | INTERFACESAFE_FOR_UNTRUSTED_DATA


/////////////////////////////////////////////////////////////////////////////
// CCtp
//
class ATL_NO_VTABLE CCtp :
    public CComObjectRootEx<CComSingleThreadModel>,
    public CComCoClass<CCtp, &CLSID_Ctp>,
    public CComControl<CCtp>,
    public IDispatchImpl<ICtp, &IID_ICtp, &LIBID_AXCTPLib>,
    public IPersistStreamInitImpl<CCtp>,
    public IOleControlImpl<CCtp>,
    public IOleObjectImpl<CCtp>,
    public IOleInPlaceActiveObjectImpl<CCtp>,
    public IViewObjectExImpl<CCtp>,
    public IOleInPlaceObjectWindowlessImpl<CCtp>,
    public IPersistPropertyBagImpl<CCtp>,
    public IObjectSafetyImpl<CCtp, dwSAFETY_OPTIONS>
{
public:
    CCtp()
    {
        m_bWindowOnly = TRUE;
        m_pCtlPanel = NULL;  .
        m_pClientDC = NULL;
        m_pAGDoc = NULL;
        m_pText = NULL;
        SetRect(&m_PageRect, 0, 0, 0, 0);
        SetRect(&m_ShadowRect, 0, 0, 0, 0);
        m_pDownloadData = NULL;
        m_dwDownloadSize = 0;
        m_dwDownloadMaxSize = 0;
        m_hBitmap = NULL;
        m_bHasFocus = false;
    }


DECLARE_REGISTRY_RESOURCEID(IDR_CTP)

BEGIN_COM_MAP(CCtp)
    COM_INTERFACE_ENTRY(ICtp)
    COM_INTERFACE_ENTRY(IDispatch)
//  COM_INTERFACE_ENTRY_IMPL(IViewObjectEx)
    COM_INTERFACE_ENTRY(IViewObjectEx)
//  COM_INTERFACE_ENTRY_IMPL_IID(IID_IViewObject2, IViewObjectEx)
    COM_INTERFACE_ENTRY_IID(IID_IViewObject2, IViewObjectEx)
//  COM_INTERFACE_ENTRY_IMPL_IID(IID_IViewObject, IViewObjectEx)
    COM_INTERFACE_ENTRY_IID(IID_IViewObject, IViewObjectEx)
//  COM_INTERFACE_ENTRY_IMPL(IOleInPlaceObjectWindowless)
    COM_INTERFACE_ENTRY(IOleInPlaceObjectWindowless)
//  COM_INTERFACE_ENTRY_IMPL_IID(IID_IOleInPlaceObject, IOleInPlaceObjectWindowless)
    COM_INTERFACE_ENTRY_IID(IID_IOleInPlaceObject, IOleInPlaceObjectWindowless)
//  COM_INTERFACE_ENTRY_IMPL_IID(IID_IOleWindow, IOleInPlaceObjectWindowless)
    COM_INTERFACE_ENTRY_IID(IID_IOleWindow, IOleInPlaceObjectWindowless)
//  COM_INTERFACE_ENTRY_IMPL(IOleInPlaceActiveObject)
    COM_INTERFACE_ENTRY(IOleInPlaceActiveObject)
//  COM_INTERFACE_ENTRY_IMPL(IOleControl)
    COM_INTERFACE_ENTRY(IOleControl)
//  COM_INTERFACE_ENTRY_IMPL(IOleObject)
```

```
    COM_INTERFACE_ENTRY(IOleObject)
//  COM_INTERFACE_ENTRY_IMPL(IPersistStreamInit)
    COM_INTERFACE_ENTRY(IPersistStreamInit)
//  COM_INTERFACE_ENTRY_IMPL_IID(IID_IPersist, IPersistPropertyBag)
    COM_INTERFACE_ENTRY_IID(IID_IPersist, IPersistPropertyBag)
//  COM_INTERFACE_ENTRY_IMPL(IPersistPropertyBag)
    COM_INTERFACE_ENTRY(IPersistPropertyBag)
//  COM_INTERFACE_ENTRY_IMPL(IObjectSafety)
    COM_INTERFACE_ENTRY(IObjectSafety)
END_COM_MAP()

BEGIN_PROPERTY_MAP(CCtp)
    PROP_ENTRY ("Fonts", 0, CLSID_NULL)
    PROP_ENTRY ("Src", 1, CLSID_NULL)
END_PROPERTY_MAP()


BEGIN_MSG_MAP(CCtp)
    MESSAGE_HANDLER(WM_CREATE, OnCreate)
    MESSAGE_HANDLER(WM_DESTROY, OnDestroy)
    MESSAGE_HANDLER(WM_ERASEBKGND, OnEraseBkgnd)
    MESSAGE_HANDLER(WM_PAINT, OnPaint)
    MESSAGE_HANDLER(WM_CHAR, OnChar)
    MESSAGE_HANDLER(WM_KEYDOWN, OnKeyDown)
    MESSAGE_HANDLER(WM_KEYUP, OnKeyUp)
    MESSAGE_HANDLER(WM_LBUTTONDBLCLK, OnLButtonDblClk)
    MESSAGE_HANDLER(WM_LBUTTONDOWN, OnLButtonDown)
    MESSAGE_HANDLER(WM_LBUTTONUP, OnLButtonUp)
    MESSAGE_HANDLER(WM_MOUSEACTIVATE, OnMouseActivate)
    MESSAGE_HANDLER(WM_MOUSEMOVE, OnMouseMove)
    MESSAGE_HANDLER(WM_TIMER, OnTimer)
    MESSAGE_HANDLER(WM_SETFOCUS, OnSetFocus)
    MESSAGE_HANDLER(WM_KILLFOCUS, OnKillFocus)
END_MSG_MAP()


// IViewObjectEx
    STDMETHOD(GetViewStatus)(DWORD *pdwStatus)
    {
        ATLTRACE(_T("IViewObjectExImpl::GetViewStatus\n"));
        *pdwStatus = 0;
        return S_OK;
    }

// IObjectSafety
    STDMETHOD(GetInterfaceSafetyOptions)(REFIID riid, DWORD *pdwSupportedOptions, DWORD *pdwEnabledO
ptions)
    {
        if (pdwSupportedOptions == NULL || pdwEnabledOptions == NULL)
            return E_POINTER;
        HRESULT hr = S_OK;
        if (riid == IID_IDispatch)
        {
            *pdwSupportedOptions = INTERFACESAFE_FOR_UNTRUSTED_CALLER | INTERFACESAFE_FOR_UNTRUSTED_
DATA;
            *pdwEnabledOptions = INTERFACESAFE_FOR_UNTRUSTED_CALLER | INTERFACESAFE_FOR_UNTRUSTED_DA
TA;
        }
        else
        {
            *pdwSupportedOptions = 0;
            *pdwEnabledOptions = 0;
            hr = E_NOINTERFACE;
        }
        return hr;
    }
    STDMETHOD(SetInterfaceSafetyOptions)(REFIID /*riid*/, DWORD /*dwOptionSetMask*/, DWORD /*dwEnabl
edOptions*/)
    {
        return S_OK;
    }

// IBindStatusCallback
```

```
    STDMETHOD(OnProgress)(ULONG /*ulProgress*/, ULONG ulProgressMax, ULONG /*ulStatusCode*/, LPCWSTR
 /*szStatusText*/)
    {
        m_dwDownloadMaxSize = ulProgressMax;

        return S_OK;
    }

// ICtp
public:
    LRESULT OnChar(UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM /*lParam*/, BOOL & /*bHandled*/);
    LRESULT OnCreate(UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM /*lParam*/, BOOL & /*bHandled*/);

    void OnData(CBindStatusCallback<CCtp> * /*pbsc*/, BYTE *pBytes, DWORD dwSize)
    {
        FileData(pBytes, dwSize);

        if (m_dwDownloadMaxSize > 0 && m_dwDownloadMaxSize == m_dwDownloadSize)
            FileEnd();
    }

    void OnFontData(CBindStatusCallback<CCtp> *pbsc, BYTE *pBytes, DWORD dwSize)
    {
        FontData(m_szFontDownload, pBytes, dwSize);
        m_dwFontDownloadSize += dwSize;

        if (m_dwDownloadMaxSize > 0 && m_dwDownloadMaxSize == m_dwFontDownloadSize)
            FontEnd(m_szFontDownload);
    }

    LRESULT OnDestroy(UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM /*lParam*/, BOOL & /*bHandled*/);
    HRESULT OnDraw(ATL_DRAWINFO &di);

    LRESULT OnEraseBkgnd(UINT /*uMsg*/, WPARAM wParam, LPARAM /*lParam*/, BOOL &bHandled)
    {
        HDC hDC = (HDC) wParam;
        HWND hParent = GetParent();

        POINT pt;
        pt.x = 0;
        pt.y = 0;
        MapWindowPoints (hParent, &pt, 1);
        OffsetWindowOrgEx (hDC, pt.x, pt.y, &pt);

        ::SendMessage(hParent, WM_ERASEBKGND, (WPARAM) hDC, 0);
        SetWindowOrgEx(hDC, pt.x, pt.y, NULL);
        bHandled = TRUE;
        return TRUE;
    }

    LRESULT OnMouseActivate(UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM /*lParam*/, BOOL & /*bHandled*/
 )
    {
        InPlaceActivate(OLEIVERB_UIACTIVATE);
        return 0;
    }

    LRESULT OnKeyDown (UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM /*lParam*/, BOOL & /*bHandled*/);
    LRESULT OnKeyUp (UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM /*lParam*/, BOOL & /*bHandled*/);
    LRESULT OnLButtonDblClk (UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM /*lParam*/, BOOL & /*bHandled*
 /);
    LRESULT OnLButtonDown (UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM /*lParam*/, BOOL & /*bHandled*/)
 ;
    LRESULT OnLButtonUp (UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM /*lParam*/, BOOL & /*bHandled*/);
    LRESULT OnMouseMove (UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM /*lParam*/, BOOL & /*bHandled*/);
    LRESULT OnKillFocus (UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM /*lParam*/, BOOL & /*bHandled*/);
    LRESULT OnSetFocus (UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM /*lParam*/, BOOL & /*bHandled*/);
    LRESULT OnTimer (UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM /*lParam*/, BOOL & /*bHandled*/);


    STDMETHOD(get_Src)(BSTR *pstrSrc)
    {
        *pstrSrc = m_bstrSrc.Copy();
        return S_OK;
    }

    STDMETHOD(put_Src)(BSTR strSrc)
    {
        USES_CONVERSION;
        m_bstrSrc = strSrc;
        LPSTR string = W2A (m_bstrSrc);

        if (string != NULL && lstrlen (string) > 0)
        {
            bool bRelativeURL = false;
            if (strchr (string, ':') == NULL)
                bRelativeURL = true;

            m_dwDownloadMaxSize = 0;
            FileStart();

            CBindStatusCallback2<CCtp>::Download(this, OnData, m_bstrSrc,
              m_spClientSite, bRelativeURL);
        }
        return S_OK;
    }

    STDMETHOD(get_Fonts)(BSTR *pstrFonts)
    {
        *pstrFonts = m_bstrFonts.Copy();
        return S_OK;
    }

    STDMETHOD(put_Fonts)(BSTR strFonts)
    {
        m_bstrFonts = strFonts;
        return S_OK;
    }

    void CreateBackPage();
    void DrawEditRect(CAGDC *pDC);
    void FileData(BYTE *pBytes, DWORD dwLen);
    void FileEnd();
    void FileStart();
    void FontData(const char *pszFontFile, BYTE *pBytes, DWORD dwLen);
    void FontEnd(const char *pszFontFile);
    void FontStart(const char *pszFontFile);
    CFontList &GetFontList()                    { return (m_FontList); }
    CAGSymImage *GetImage(int nID);
    CAGText *GetText()                          { return (m_pText); }
    bool HasFocus()                             { return (m_bHasFocus); }
    void NewPage();

    void StartDownloadFont(const char *pszFontName)
    {
        USES_CONVERSION;
        char szFontURL[_MAX_PATH];
        LPSTR string = W2A (m_bstrFonts);
        lstrcpy (szFontURL, string);
        lstrcat (szFontURL, pszFontName);
        if (string != NULL && lstrlen (string) > 0)
        {
            bool bRelativeURL = false;
            if (strchr (string, ':') == NULL)
                bRelativeURL = true;

            m_dwDownloadMaxSize = 0;
            m_dwFontDownloadSize = 0;
            FontStart(pszFontName);
            lstrcpy (m_szFontDownload, pszFontName);

            CComBSTR bstr = szFontURL;
            CBindStatusCallback2<CCtp>::Download(this, OnFontData, bstr,
              m_spClientSite, bRelativeURL);
        }
    }
    void StartEdit(CAGSymText *pText, POINT Pt, bool bClick);
    void StopEdit();

protected:
    CCtlPanel           *m_pCtlPanel;
    CAGClientDC         *m_pClientDC;
    CAGDoc              *m_pAGDoc;
    CAGSymText          *m_pText;
    RECT                m_PageRect;
    RECT                m_ShadowRect;
    CAGMatrix           m_ViewMatrix;
    CComBSTR            m_bstrSrc;
    CComBSTR            m_bstrFonts;
    BYTE                *m_pDownloadData;
    DWORD               m_dwDownloadSize;
    DWORD               m_dwDownloadMaxSize;
    HBITMAP             m_hBitmap;
    CFontList           m_FontList;
    FONTDOWNLOADARRAY   m_FontDownloadArray;
    char                m_szFontDownload[_MAX_FNAME];
    DWORD               m_dwFontDownloadSize;
    bool                m_bHasFocus;
};

#endif //__CTP_H_
```

```
//============================================================================//
//============================================================================//
#include "stdafx.h"
#include "Ctp.h"
#include "AGDoc.h"
#include "AGPage.h"
#include "AGLayer.h"
#include "AGDC.h"
#include "Font.h"

#include <strstrea.h>
#include <scselect.h>

#define TIMER_TEXT  1

static HHOOK g_hHook=NULL;
static CCtp *g_pThis;


//----------------------------------------------------------------------------//
//----------------------------------------------------------------------------//
LRESULT CALLBACK CtpKeyboardProc(int nCode, WPARAM wParam, LPARAM lParam)
{
    if (nCode < 0 || !g_pThis->HasFocus())
        return ::CallNextHookEx(g_hHook, nCode, wParam, lParam);

    switch (wParam)
    {
        case VK_LEFT:
        case VK_RIGHT:
        case VK_UP:
        case VK_DOWN:
        case VK_HOME:
        case VK_END:
        case VK_PRIOR:
        case VK_NEXT:
        {
            int bHandled;
            if (lParam & 0x80000000)
                g_pThis->OnKeyUp(0, wParam, 0, bHandled);
            else
                g_pThis->OnKeyDown(0, wParam, 0, bHandled);
            return (1);
            break;
        }

        case VK_ESCAPE:
        case VK_TAB:
        case VK_BACK:
        {
            int bHandled;
            if ((lParam & 0x80000000) == 0)
                g_pThis->OnChar(0, wParam, 0, bHandled);
            return (1);
            break;
        }
    }

    return ::CallNextHookEx(g_hHook, nCode, wParam, lParam);
}


//----------------------------------------------------------------------------//
//----------------------------------------------------------------------------//
void CCtp::CreateBackPage()
{
    CAGPage *pAGPage = m_pAGDoc->GetPage(4);
    SIZE sizePage;
    pAGPage->GetPageSize(&sizePage);

    if (pAGPage->GetNumLayers() != 2)
    {
        CAGLayer *pAGLayer = new CAGLayer();
        pAGPage->AppendLayer(pAGLayer);
    }

    CAGSymImage *pAGLogo = GetImage(IDR_AGLOGO);
    CAGSymImage *pCPLogo = GetImage(IDR_CPLOGO);

    RECT AGLogoRect;
    RECT ImageRect = pAGLogo->GetDestRect();
    AGLogoRect.left = (sizePage.cx - WIDTH(ImageRect)) / 2;
    AGLogoRect.top = (sizePage.cy - HEIGHT(ImageRect)) / 2;
    AGLogoRect.right = AGLogoRect.left + WIDTH(ImageRect);
    AGLogoRect.bottom = AGLogoRect.top + HEIGHT(ImageRect);

    RECT CPLogoRect;
    ImageRect = pCPLogo->GetDestRect();
    CPLogoRect.left = (APP_RESOLUTION / 16);
    CPLogoRect.bottom = sizePage.cy - (APP_RESOLUTION / 16);
    CPLogoRect.top = CPLogoRect.bottom - HEIGHT(ImageRect);
    CPLogoRect.right = CPLogoRect.left + WIDTH(ImageRect);

    char szMsg[] = "Created\njust for you\nby\nsender's name";
    char szCopyright[] = "©AGC, Inc.";

    RECT MsgRect;
    RECT CopyrightRect;
    CopyrightRect.left = sizePage.cx - ((APP_RESOLUTION * 15) / 10);
    CopyrightRect.top = sizePage.cy - (APP_RESOLUTION / 2);
    CopyrightRect.right = sizePage.cx - (APP_RESOLUTION / 16);
    CopyrightRect.bottom = sizePage.cy - (APP_RESOLUTION / 16);

    LOGFONT MsgFont;
    memset(&MsgFont, 0, sizeof(MsgFont));
    MsgFont.lfWeight = 400;
    lstrcpy(MsgFont.lfFaceName, "CAC Futura Casual");

    LOGFONT CopyrightFont;
    memset(&CopyrightFont, 0, sizeof(CopyrightFont));
    CopyrightFont.lfWeight = 400;
    lstrcpy(CopyrightFont.lfFaceName, "Arial");

    switch (m_pAGDoc->GetDocType())
    {
        case DOC_CARDHV:
        {
            MsgFont.lfHeight = -(14 * APP_RESOLUTION / 72);
            MsgRect.left = APP_RESOLUTION / 4;
            MsgRect.top = APP_RESOLUTION / 2;
            MsgRect.right = sizePage.cx - (APP_RESOLUTION / 4);
            MsgRect.bottom = APP_RESOLUTION * 2;

            CopyrightFont.lfHeight = -(12 * APP_RESOLUTION / 72);
            break;
        }

        case DOC_CARDHH:
        {
            MsgFont.lfHeight = -(14 * APP_RESOLUTION / 72);
            MsgRect.left = APP_RESOLUTION / 4;
            MsgRect.top = APP_RESOLUTION / 4;
            MsgRect.right = sizePage.cx - (APP_RESOLUTION / 4);
            MsgRect.bottom = (APP_RESOLUTION * 175) / 100;

            CopyrightFont.lfHeight = -(12 * APP_RESOLUTION / 72);

            AGLogoRect.top += (APP_RESOLUTION / 2);
            AGLogoRect.bottom += (APP_RESOLUTION / 2);
            break;
        }

        case DOC_CARDFV:
        {
            MsgFont.lfHeight = -(11 * APP_RESOLUTION / 72);
            MsgRect.left = APP_RESOLUTION / 8;
            MsgRect.top = APP_RESOLUTION / 4;
            MsgRect.right = sizePage.cx - (APP_RESOLUTION / 8);
            MsgRect.bottom = (APP_RESOLUTION * 15) / 10;

            CopyrightFont.lfHeight = -(8 * APP_RESOLUTION / 72);

            AGLogoRect.top += (APP_RESOLUTION / 8);
            AGLogoRect.bottom += (APP_RESOLUTION / 8);

            CAGMatrix Matrix;
            Matrix.Scale(.75, .75,
                        ((AGLogoRect.left + AGLogoRect.right) / 2),
                        ((AGLogoRect.top + AGLogoRect.bottom) / 2));
            pAGLogo->SetMatrix(Matrix);

            Matrix.Unity();
            Matrix.Scale(.70, .70, CPLogoRect.left, CPLogoRect.bottom);
            pCPLogo->SetMatrix(Matrix);
            break;
        }

        case DOC_CARDFH:
        {
            MsgFont.lfHeight = -(11 * APP_RESOLUTION / 72);
            MsgRect.left = APP_RESOLUTION / 8;
            MsgRect.top = APP_RESOLUTION / 16;
            MsgRect.right = sizePage.cx - (APP_RESOLUTION / 8);
            MsgRect.bottom = (APP_RESOLUTION * 125) / 100;

            CopyrightFont.lfHeight = -(8 * APP_RESOLUTION / 72);

            AGLogoRect.top += (APP_RESOLUTION / 5);
            AGLogoRect.bottom += (APP_RESOLUTION / 5);

            CAGMatrix Matrix;
            Matrix.Scale(.75, .75,
                        ((AGLogoRect.left + AGLogoRect.right) / 2),
                        ((AGLogoRect.top + AGLogoRect.bottom) / 2));
            pAGLogo->SetMatrix(Matrix);

            Matrix.Unity();
            Matrix.Scale(.70, .70, CPLogoRect.left, CPLogoRect.bottom);
            pCPLogo->SetMatrix(Matrix);
            break;
        }
    }

    CAGSpec MsgSpec(MsgFont, RGB(0, 0, 0), eRagCentered);
    CAGSpec CopyrightSpec(CopyrightFont, RGB(0, 0, 0), eRagLeft);

    int nSpecOffset = 0;

    CAGSymText *pAGSymText = new CAGSymText();
    pAGSymText->Create(MsgRect);
    pAGSymText->SetVertJust(eVertCentered);
    pAGSymText->SetText(szMsg, lstrlen(szMsg), 1, &MsgSpec, &nSpecOffset);

    CAGLayer *pAGLayer = pAGPage->GetLayer(2);
    pAGLayer->AppendSymbol(pAGSymText);

    pAGLayer = pAGPage->GetLayer(1);

    pAGLogo->SetDestRect(AGLogoRect);
    pAGLayer->AppendSymbol(pAGLogo);

    pCPLogo->SetDestRect(CPLogoRect);
    pAGLayer->AppendSymbol(pCPLogo);

    pAGSymText = new CAGSymText();
    pAGSymText->Create(CopyrightRect);
    pAGSymText->SetVertJust(eVertBottom);
    pAGSymText->SetText(szCopyright, lstrlen (szCopyright), 1, &CopyrightSpec,
                        &nSpecOffset);
    pAGLayer->AppendSymbol(pAGSymText);
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CCtp::DrawEditRect(CAGDC *pDC)
{
    if (m_pText)
    {
        RECT DestRect = m_pText->GetDestRect();
        RECT Offset = {-1, -1, 1, 1};
        pDC->DPAtoLPA(&Offset);
        DestRect.left += Offset.left;
        DestRect.right += Offset.right;
        DestRect.top += Offset.top;
        DestRect.bottom += Offset.bottom;

        POINT Pts[5];
        Pts[0].x = Pts[1].x = Pts[4].x = DestRect.left;
        Pts[2].x = Pts[3].x = DestRect.right;
        Pts[0].y = Pts[3].y = Pts[4].y = DestRect.top;
        Pts[1].y = Pts[2].y = DestRect.bottom;
        pDC->LPtoDP(Pts, 5);

        int PrevROP = ::SetROP2(pDC->GetHDC(), R2_NOTXORPEN);
        HBRUSH hOldBrush = (HBRUSH) ::SelectObject(pDC->GetHDC(),
                                              ::GetStockObject(NULL_BRUSH));
        HPEN hOldPen = (HPEN) ::SelectObject(pDC->GetHDC(),
                                              ::GetStockObject(BLACK_PEN));
        ::Polyline(pDC->GetHDC(), Pts, 5);
        ::SelectObject(pDC->GetHDC(), hOldBrush);
        ::SelectObject(pDC->GetHDC(), hOldPen);
        ::SetROP2(pDC->GetHDC(), PrevROP);
    }
}

//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CCtp::FileData(BYTE *pBytes, DWORD dwLen)
{
    if (m_pDownloadData == NULL)
        m_pDownloadData = (BYTE *)malloc(dwLen);
    else
        m_pDownloadData = (BYTE *)realloc(m_pDownloadData, m_dwDownloadSize + dwLen);

    memcpy(m_pDownloadData + m_dwDownloadSize, pBytes, dwLen);
    m_dwDownloadSize += dwLen;
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CCtp::FileEnd()
{
    if (m_pAGDoc)
    {
        delete m_pAGDoc;
        m_pAGDoc = NULL;
    }

    istrstream input((char *)m_pDownloadData, m_dwDownloadSize);

    m_pAGDoc = new CAGDoc();
    if (! m_pAGDoc->Read(input))
    {
        delete m_pAGDoc;
        m_pAGDoc = NULL;
        free(m_pDownloadData);
        m_pDownloadData = NULL;
        m_dwDownloadSize = 0;
    }
    else
    {
        LOGFONTARRAY m_DocFonts;
        m_pAGDoc->GetFonts(m_DocFonts);
        m_FontList.CheckFonts(m_DocFonts, m_FontDownloadArray);
        if (m_FontDownloadArray.empty())
        {
            CreateBackPage();
            free(m_pDownloadData);
            m_pDownloadData = NULL;
            m_dwDownloadSize = 0;
        }
        else
        {
            delete m_pAGDoc;
            m_pAGDoc = NULL;

#if 0
            int nDownload = m_FontDownloadArray.size();
            for (int i = 0; i < nDownload; i++)
            {
                StartDownloadFont(m_FontDownloadArray[i].szFontFile);
            }
#else
            if (! m_FontDownloadArray.empty())
                StartDownloadFont(m_FontDownloadArray[0].szFontFile);
#endif
        }
    }

    if (m_hWnd && m_pAGDoc)
    {
        m_pCtlPanel->SetDoc(m_pAGDoc);
        m_pCtlPanel->ShowWindow(SW_SHOW);
        NewPage();
        ::InvalidateRect(GetParent(), NULL, TRUE);
        Invalidate();
    }
}

//-------------------------------------------------------------------------//
//-------------------------------------------------------------------------//
void CCtp::FileStart()
{
    if (m_pDownloadData)
        free(m_pDownloadData);
    m_pDownloadData = NULL;
    m_dwDownloadSize = 0;
}


//-------------------------------------------------------------------------//
//-------------------------------------------------------------------------//
void CCtp::FontData(const char *pszFontFile, BYTE *pBytes, DWORD dwLen)
{
    int nDownload = m_FontDownloadArray.size();
    for (int i = 0; i < nDownload; i++)
    {
        if (lstrcmpi(pszFontFile, m_FontDownloadArray[i].szFontFile) == 0)
        {
            if (m_FontDownloadArray[i].pDownloadData == NULL)
                m_FontDownloadArray[i].pDownloadData = (BYTE *)malloc(dwLen);
            else
            {
                m_FontDownloadArray[i].pDownloadData = (BYTE *)realloc(
                                            m_FontDownloadArray[i].pDownloadData,
                                            m_FontDownloadArray[i].dwDownloadSize + dwLen);
            }

            memcpy(m_FontDownloadArray[i].pDownloadData +
                    m_FontDownloadArray[i].dwDownloadSize, pBytes, dwLen);
            m_FontDownloadArray[i].dwDownloadSize += dwLen;
            break;
        }
    }
}


//-------------------------------------------------------------------------------//
//-------------------------------------------------------------------------------//
void CCtp::FontEnd(const char *pszFontFile)
{
    int nDownload = m_FontDownloadArray.size();
    for (int i = 0; i < nDownload; i++)
    {
        if (lstrcmpi(pszFontFile, m_FontDownloadArray[i].szFontFile) == 0)
        {
            if (m_FontDownloadArray[i].pDownloadData)
            {
                m_FontList.InstallFont(m_FontDownloadArray[i].pDownloadData,
                  m_FontDownloadArray[i].dwDownloadSize,
                  m_FontDownloadArray[i].szFullName, pszFontFile);

                free(m_FontDownloadArray[i].pDownloadData);
            }
            m_FontDownloadArray[i].pDownloadData = NULL;
            m_FontDownloadArray[i].dwDownloadSize = 0;

            m_FontDownloadArray.erase(m_FontDownloadArray.begin() + i);

            if (m_FontDownloadArray.empty() && m_pDownloadData)
            {
                m_FontList.InitFontArray();
                if (m_pCtlPanel)
                    m_pCtlPanel->SetFonts();

                istrstream input((char *)m_pDownloadData, m_dwDownloadSize);
                m_pAGDoc = new CAGDoc();
                if (! m_pAGDoc->Read(input))
                {
                    delete m_pAGDoc;
                    m_pAGDoc = NULL;
                    free(m_pDownloadData);
                    m_pDownloadData = NULL;
                    m_dwDownloadSize = 0;
                }
                else
                {
                    CreateBackPage();
                    free(m_pDownloadData);
                    m_pDownloadData = NULL;
                    m_dwDownloadSize = 0;
                }

                if (m_hWnd && m_pAGDoc)
                {
                    m_pCtlPanel->SetDoc(m_pAGDoc);
                    m_pCtlPanel->ShowWindow(SW_SHOW);
                    NewPage();
                    ::InvalidateRect(GetParent(), NULL, TRUE);
                    Invalidate();
                }
            }
 #if 1
            else if (! m_FontDownloadArray.empty())
            {
                StartDownloadFont(m_FontDownloadArray[0].szFontFile);
            }
 #endif
            break;
        }
    }
}
```

```
//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CCtp::FontStart(const char *pszFontFile)
{
    int nDownload = m_FontDownloadArray.size();
    for (int i = 0; i < nDownload; i++)
    {
        if (lstrcmpi(pszFontFile, m_FontDownloadArray[i].szFontFile) == 0)
        {
            if (m_FontDownloadArray[i].pDownloadData)
                free(m_FontDownloadArray[i].pDownloadData);
            m_FontDownloadArray[i].pDownloadData = NULL;
            m_FontDownloadArray[i].dwDownloadSize = 0;
            break;
        }
    }
}

//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
CAGSymImage *CCtp::GetImage(int nID)
{
    CAGSymImage *pAGSymImage = NULL;
    HGLOBAL     hSymImage = NULL;
    HRSRC       hResource;

    if ((hResource = ::FindResource(_Module.GetResourceInstance(),
        MAKEINTRESOURCE(nID), "AGIMAGE")) != NULL)
    {
        hSymImage = ::LoadResource(_Module.GetResourceInstance(), hResource);
    }

    if (hSymImage)
    {
        istrstream input((char *)::LockResource(hSymImage),
          ::SizeofResource(_Module.GetResourceInstance(), hResource));
        CAGDocIO DocIO(&input);

        pAGSymImage = new CAGSymImage();
        if (! pAGSymImage->Read(&DocIO))
        {
            delete pAGSymImage;
            pAGSymImage = NULL;
        }

        DocIO.Close();
    }
    return (pAGSymImage);
}

//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CCtp::NewPage()
{
    if (m_pText && m_pText->IsEditing())
        StopEdit();

    RECT RepaintRect;
    ::UnionRect(&RepaintRect, &m_PageRect, &m_ShadowRect);

    SIZE sizeShadow = { (APP_RESOLUTION * 4) / 100,
                        (APP_RESOLUTION * 4) / 100 };

    RECT WndRect, DlgRect;
    GetClientRect(&WndRect);
    m_pCtlPanel->GetClientRect(&DlgRect);
    WndRect.left += 10;
    WndRect.top += 2;
    WndRect.right -= (WIDTH(DlgRect) + 10);
    WndRect.bottom -= 2;
    m_pClientDC->DPtoVP(&WndRect);
    WndRect.right -= sizeShadow.cx;
    WndRect.bottom -= sizeShadow.cy;

    SIZE sizePage;
    m_pAGDoc->GetCurrentPage()->GetPageSize(&sizePage);

    ::SetRect(&m_PageRect, 0, 0, sizePage.cx, sizePage.cy);
    m_ViewMatrix.ScaleAndCenter(WndRect, m_PageRect);

    m_ViewMatrix.m_31 = WndRect.left;
    m_ViewMatrix.m_32 = WndRect.top;

    m_pClientDC->SetViewingMatrix(m_ViewMatrix);
    m_pClientDC->MPtoDP(&m_PageRect);

    m_pCtlPanel->SetWindowPos(NULL, m_PageRect.right + 10, 0, 0, 0, SWP_NOSIZE | SWP_NOZORDER);


    m_ShadowRect = m_PageRect;
    m_pClientDC->VPAtoDPA((POINT *)&sizeShadow, 1);
    ::OffsetRect(&m_ShadowRect, sizeShadow.cx, sizeShadow.cy);

    CAGMatrix TempMatrix((double)WIDTH(m_PageRect) / (double)sizePage.cx,
                         0, 0, (double)HEIGHT(m_PageRect) / (double)sizePage.cy, 1, 1);


    ::InflateRect(&m_PageRect, 1, 1);
    ::UnionRect(&RepaintRect, &RepaintRect, &m_PageRect);
    ::UnionRect(&RepaintRect, &RepaintRect, &m_ShadowRect);

    if (m_hBitmap)
    {
        ::DeleteObject(m_hBitmap);
        m_hBitmap = NULL;
    }

    HDC hMemDC = ::CreateCompatibleDC(m_pClientDC->GetHDC());
    m_hBitmap = ::CreateCompatibleBitmap(m_pClientDC->GetHDC(), WIDTH(m_PageRect), HEIGHT(m_PageRect
));
    HBITMAP hOldBitmap = (HBITMAP)::SelectObject(hMemDC, m_hBitmap);

    HBRUSH hbrOld = (HBRUSH)::SelectObject(hMemDC, GetStockObject(WHITE_BRUSH));
    HPEN hpenOld = (HPEN)::SelectObject(hMemDC, GetStockObject(BLACK_PEN));
    ::Rectangle(hMemDC, 0, 0, WIDTH(m_PageRect), HEIGHT(m_PageRect));
    ::SelectObject(hMemDC, hbrOld);
    ::SelectObject(hMemDC, hpenOld);

    CAGDC dc (hMemDC);
    dc.SetDeviceMatrix(TempMatrix);
    m_pAGDoc->GetCurrentPage()->GetLayer(1)->Draw(dc);

    ::SelectObject(hMemDC, hOldBitmap);
    ::DeleteDC(hMemDC);


    InvalidateRect(&RepaintRect);
    UpdateWindow();

    CAGLayer *pLayer = m_pAGDoc->GetCurrentPage()->GetLayer(2);
    CAGSymText *pText = (CAGSymText *)pLayer->FindFirstSymbolByType(ST_TEXT);
    if (pText)
    {
        const RECT &DestRect = pText->GetDestRect();
        POINT pt = {DestRect.left, DestRect.top};
        StartEdit(pText, pt, false);
        SetFocus();
    }

    m_pCtlPanel->UpdateControls(m_pText);
}
```

```
//-----------------------------------------------------------------------------//
//-----------------------------------------------------------------------------//
LRESULT CCtp::OnCreate(UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM /*lParam*/,
                        BOOL & /*bHandled*/)
{
    DWORD ClassStyle = ::GetClassLong(m_hWnd, GCL_STYLE);
    ::SetClassLong(m_hWnd, GCL_STYLE, ClassStyle | CS_DBLCLKS);

    m_pClientDC = new CAGClientDC(m_hWnd);
    m_pCtlPanel = new CCtlPanel(this);
    m_pCtlPanel->Create(m_hWnd);

    if (m_pAGDoc)
    {
        NewPage();
        m_pCtlPanel->SetDoc(m_pAGDoc);
        m_pCtlPanel->ShowWindow(SW_SHOW);
    }

    g_hHook = ::SetWindowsHookEx(WH_KEYBOARD,
      reinterpret_cast<HOOKPROC>(CtpKeyboardProc), NULL, GetCurrentThreadId());
    g_pThis = this;

    return 0;
}

//-----------------------------------------------------------------------------//
//-----------------------------------------------------------------------------//
LRESULT CCtp::OnDestroy(UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM /*lParam*/,
                        BOOL & /*bHandled*/)
{
    ::UnhookWindowsHookEx(g_hHook);

    if (m_pCtlPanel)
    {
        m_pCtlPanel->DestroyWindow();
        delete m_pCtlPanel;
        m_pCtlPanel = NULL;
    }
    if (m_pAGDoc)
    {
        delete m_pAGDoc;
        m_pAGDoc = NULL;
    }
    if (m_pClientDC)
    {
        delete m_pClientDC;
        m_pClientDC = NULL;
    }

    if (m_hBitmap)
    {
        ::DeleteObject(m_hBitmap);
        m_hBitmap = NULL;
    }
    return 0;
}

//-----------------------------------------------------------------------------//
//-----------------------------------------------------------------------------//
HRESULT CCtp::OnDraw(ATL_DRAWINFO &di)
{
    BOOL bUserMode;
    GetAmbientUserMode(bUserMode);
    if (! bUserMode)
    {
        RECT &rc = *(RECT*)di.prcBounds;

        ::FillRect(di.hdcDraw, &rc, (HBRUSH)GetStockObject(WHITE_BRUSH));
        BSTR bstr;
        if (SUCCEEDED(GetAmbientDisplayName(bstr)))
        {
            USES_CONVERSION;
            ::DrawText(di.hdcDraw, OLE2A(bstr), -1, &rc, DT_TOP | DT_SINGLELINE);
        }
    }
    else if (m_pAGDoc)
    {
        CAGDC dc(di.hdcDraw);
        RECT r = m_ShadowRect;
        r.left = m_PageRect.right;
        ::FillRect(di.hdcDraw, &r, (HBRUSH)GetStockObject(BLACK_BRUSH));
        r.left = m_ShadowRect.left;
        r.top = m_PageRect.bottom;
        ::FillRect(di.hdcDraw, &r, (HBRUSH)GetStockObject(BLACK_BRUSH));


        HDC hMemDC = ::CreateCompatibleDC(di.hdcDraw);
        CAGDC dcTemp(hMemDC);
        HBITMAP hOldBitmap = (HBITMAP) ::SelectObject(hMemDC, m_hBitmap);
        ::BitBlt(di.hdcDraw, m_PageRect.left, m_PageRect.top,
                    WIDTH(m_PageRect), HEIGHT(m_PageRect), hMemDC, 0, 0, SRCCOPY);
        ::SelectObject(hMemDC, hOldBitmap);
        ::DeleteDC(hMemDC);

        dc.SetViewingMatrix(m_ViewMatrix);
        m_pAGDoc->GetCurrentPage()->GetLayer(2)->Draw(dc);


        if (m_pText && m_pText->IsEditing())
        {
            dc.PushModelingMatrix(m_pText->GetMatrix());
            m_pText->DrawSelection(dc);
            DrawEditRect(&dc);
            dc.PopModelingMatrix();
        }

    }
    else
    {
        RECT &rc = *(RECT*)di.prcBounds;

        ::SetTextColor(di.hdcDraw, RGB(0, 0, 0));
        ::SetBkMode(di.hdcDraw, TRANSPARENT);

        RECT rp;
        HWND hParent = GetParent();
        ::GetClientRect(hParent, &rp);
        ::MapWindowPoints(hParent, m_hWnd, (POINT *)&rp, 2);
        RECT r;
        ::IntersectRect(&r, &rc, &rp);

        ::DrawText(di.hdcDraw, "Preparing to edit card.  Please Wait...", -1,
                    &r, DT_CENTER | DT_VCENTER | DT_SINGLELINE);
    }

    return S_OK;
}

//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
LRESULT CCtp::OnChar(UINT /*uMsg*/, WPARAM wParam, LPARAM /*lParam*/,
                        BOOL & /*bHandled*/)
{
    UINT nChar = wParam;

    if (m_pText && m_pText->IsEditing())
    {
        if (nChar == VK_ESCAPE)
            StopEdit();
        else
```

```
            m_pText->OnChar(nChar);
    }

    return 0;
}


//------------------------------------------------------------------------------//
//------------------------------------------------------------------------------//
LRESULT CCtp::OnKeyDown(UINT /*uMsg*/, WPARAM wParam, LPARAM /*lParam*/,
                        BOOL & /*bHandled*/)
{
    UINT nChar = wParam;

    if (m_pText && m_pText->IsEditing())
    {
        m_pText->OnKeyDown(nChar);
        m_pCtlPanel->UpdateControls(m_pText);
    }

    return 0;
}


//------------------------------------------------------------------------------//
//------------------------------------------------------------------------------//
LRESULT CCtp::OnKeyUp(UINT /*uMsg*/, WPARAM wParam, LPARAM /*lParam*/,
                        BOOL & /*bHandled*/)
{
    UINT nChar = wParam;

    if (m_pText && m_pText->IsEditing())
        m_pText->OnKeyUp(nChar);

    return 0;


//------------------------------------------------------------------------------//
//------------------------------------------------------------------------------//
LRESULT CCtp::OnLButtonDblClk(UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM lParam,
                                BOOL & /*bHandled*/)

    if (m_pText && m_pText->IsEditing())
    {
        POINT pt = { LOWORD(lParam), HIWORD(lParam) };
        m_pClientDC->DPtoLP(&pt, 1);
        m_pText->OnLButtonDblClk(pt);
        m_pCtlPanel->UpdateControls(m_pText);
    }
    return 0;
}


//------------------------------------------------------------------------------//
//------------------------------------------------------------------------------//
LRESULT CCtp::OnLButtonDown(UINT /*uMsg*/, WPARAM wParam, LPARAM lParam,
                            BOOL & /*bHandled*/)
{
    if (m_pAGDoc)
    {
        SetCapture();

        POINT pt = { LOWORD(lParam), HIWORD(lParam) };
        m_pClientDC->DPtoMP(&pt, 1);
        CAGSym *pSym = m_pAGDoc->GetCurrentPage()->GetLayer(2)->FindSymbolByPoint(pt, ST_TEXT);
        if (pSym)
        {
            pt.x = LOWORD(lParam);
            pt.y = HIWORD(lParam);
            m_pClientDC->DPtoLP(&pt, 1);

            if (pSym != m_pText)
            {
                if (m_pText && m_pText->IsEditing())
                    StopEdit();

                StartEdit((CAGSymText *)pSym, pt, true);
            }
            else if (m_pText->IsEditing())
                m_pText->OnLButtonDown(pt, (wParam & MK_SHIFT) != 0);

            SetFocus();
            m_pCtlPanel->UpdateControls(m_pText);
        }
        else if (m_pText)
        {
            if (m_pText->IsEditing())
                StopEdit();
            m_pText = NULL;
        }
    }
    return 0;
}


//------------------------------------------------------------------------//
//------------------------------------------------------------------------//
LRESULT CCtp::OnLButtonUp(UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM lParam,
                            BOOL & /*bHandled*/)
{
    if (m_pText && m_pText->IsEditing())
    {
        POINT pt = { LOWORD(lParam), HIWORD(lParam) };
        m_pClientDC->DPtoLP(&pt, 1);
        m_pText->OnLButtonUp(pt);
        if (! m_pText->GetSelection()->IsSliverCursor())
            m_pCtlPanel->UpdateControls(m_pText);
    }

    ReleaseCapture();

    return 0;
}


//------------------------------------------------------------------------//
//------------------------------------------------------------------------//
LRESULT CCtp::OnMouseMove(UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM lParam,
                            BOOL & /*bHandled*/)
{
    if (m_pText && m_pText->IsEditing())
    {
        POINT pt = { LOWORD(lParam), HIWORD(lParam) };
        m_pClientDC->DPtoLP(&pt, 1);
        m_pText->OnMouseMove(pt);
    }
    return 0;
}


//------------------------------------------------------------------------//
//------------------------------------------------------------------------//
LRESULT CCtp::OnKillFocus(UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM /* lParam */,
                            BOOL & /*bHandled*/)
{
    if (m_pText && m_pText->IsEditing())
    {
        KillTimer(TIMER_TEXT);

        if (m_pText->GetSelection()->IsSliverCursor())
            m_pText->ShowSelection(false);
    }
    m_bHasFocus = false;

    return 0;
```

```
}


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
LRESULT CCtp::OnSetFocus(UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM /* lParam */,
                            BOOL & /*bHandled*/)
{
    if (m_pText && m_pText->IsEditing())
    {
        if (m_pText->GetSelection()->IsSliverCursor())
            m_pText->ShowSelection(true);
//      SetTimer(TIMER_TEXT, 500, NULL);
        SetTimer(TIMER_TEXT, 500);
        m_bHasFocus = true;
    }

    return 0;
}


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
LRESULT CCtp::OnTimer(UINT /*uMsg*/, WPARAM wParam, LPARAM /* lParam */,
                        BOOL & /*bHandled*/)
{
    if (wParam == TIMER_TEXT)
    {
        if (m_pText && m_pText->IsEditing())
            m_pText->BlinkCursor();
        else
            KillTimer(TIMER_TEXT);
    }

    return 0;
}


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
void CCtp::StartEdit(CAGSymText *pText, POINT Pt, bool bClick)
{
    m_pText = pText;

    m_pClientDC->PushModelingMatrix(m_pText->GetMatrix());

    m_pText->Edit(m_pClientDC, Pt.x, Pt.y, bClick);
    DrawEditRect(m_pClientDC);
//  SetTimer(TIMER_TEXT, 500, NULL);
    SetTimer(TIMER_TEXT, 500);
}


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
void CCtp::StopEdit()
{
    if (m_pText && m_pText->IsEditing())
    {
        KillTimer(TIMER_TEXT);
        m_pText->EndEdit();
        DrawEditRect(m_pClientDC);
        m_pClientDC->PopModelingMatrix();
    }
    m_pText = NULL;
    m_pCtlPanel->UpdateControls(NULL);
}
```

```
#ifndef __CTLPANEL_H_
#define __CTLPANEL_H_

#include "resource.h"
#include "AGDoc.h"
#include "AGText.h"
#include "Font.h"

class CCtp;

/////////////////////////////////////////////////////////////////////////////
// CCtlPanel
class CCtlPanel :
    public CDialogImpl<CCtlPanel>
{
public:
    CCtlPanel (CCtp *pMainWnd);
    ~CCtlPanel ();

    enum { IDD = IDD_CTLPANEL };

BEGIN_MSG_MAP (CCtlPanel)
    MESSAGE_HANDLER (WM_INITDIALOG, OnInitDialog)
    MESSAGE_HANDLER (WM_CTLCOLORDLG, OnCtlColorDlg)
    MESSAGE_HANDLER (WM_CTLCOLORSTATIC, OnCtlColorStatic)
    MESSAGE_HANDLER (WM_MEASUREITEM, OnMeasureItem)
    MESSAGE_HANDLER (WM_DRAWITEM, OnDrawItem)
    COMMAND_ID_HANDLER (IDC_PAGE1, OnPage)
    COMMAND_ID_HANDLER (IDC_PAGE2, OnPage)
    COMMAND_ID_HANDLER (IDC_PAGE3, OnPage)
    COMMAND_ID_HANDLER (IDC_PAGE4, OnPage)
    COMMAND_ID_HANDLER (IDC_LEFT, OnHorzJust)
    COMMAND_ID_HANDLER (IDC_CENTER, OnHorzJust)
    COMMAND_ID_HANDLER (IDC_RIGHT, OnHorzJust)
    COMMAND_ID_HANDLER (IDC_PRINT, OnPrint)
    COMMAND_ID_HANDLER (IDC_FONT, OnFont)
    COMMAND_ID_HANDLER (IDC_PTSIZE, OnPtSize)
    COMMAND_ID_HANDLER (IDC_COLOR, OnColor)

END_MSG_MAP ()

    LRESULT OnInitDialog (UINT uMsg, WPARAM wParam, LPARAM lParam, BOOL& bHandled);
    LRESULT OnCtlColorDlg (UINT uMsg, WPARAM wParam, LPARAM lParam, BOOL& bHandled);
    LRESULT OnCtlColorStatic (UINT uMsg, WPARAM wParam, LPARAM lParam, BOOL& bHandled);
    LRESULT OnMeasureItem (UINT uMsg, WPARAM wParam, LPARAM lParam, BOOL& bHandled);
    LRESULT OnDrawItem (UINT uMsg, WPARAM wParam, LPARAM lParam, BOOL& bHandled);
    LRESULT OnPage (WORD wNotifyCode, WORD wID, HWND hWndCtl, BOOL& bHandled);
    LRESULT OnPrint (WORD wNotifyCode, WORD wID, HWND hWndCtl, BOOL& bHandled);
    LRESULT OnHorzJust (WORD wNotifyCode, WORD wID, HWND hWndCtl, BOOL& bHandled);
    LRESULT OnFont (WORD wNotifyCode, WORD wID, HWND hWndCtl, BOOL& bHandled);
    LRESULT OnPtSize (WORD wNotifyCode, WORD wID, HWND hWndCtl, BOOL& bHandled);
    LRESULT OnColor (WORD wNotifyCode, WORD wID, HWND hWndCtl, BOOL& bHandled);

    HWND Create (HWND hWndParent);
    FONTARRAY &GetFontArray ();
    void SetDoc (CAGDoc *pAGDoc);
    void SetFonts ();
    void UpdateControls (const CAGText *pText);

protected:
    CCtp        *m_pMainWnd;
    CAGDoc      *m_pAGDoc;
    HGLOBAL     m_hDevMode;
    HGLOBAL     m_hDevNames;
    int         m_PageMap[4];
    int         m_nFontHeight;
    HGLOBAL     m_hDlg;
};

#endif //__CTLPANEL_H_
```

```
//==========================================================================//
//==========================================================================//
#include "stdafx.h"
#include "CtlPanel.h"
#include "Ctp.h"
#include "WaitDlg.h"
#include "AGPage.h"
#include "DblSide.h"

#define DUPLEX_SUBKEY   "Software\\American Greetings\\Create and Print\\Duplex Printing"

//--------------------------------------------------------------------------//
//     Duplex 1       Duplex 2       Duplex 3       Duplex 4               //
//    Front  Back    Front  Back    Front  Back    Front  Back             //
//                                                                          //
//    ^ ^ ^             ^              ^    ^  ^      ^                     //
//    | | |             |              |    |  |      |                     //
//                    |   |                                |  |             //
//                    v   v                                v  v             //
//                                                                          //
//                                                                          //
// 0 = toward   1 = away from                                               //
//--------------------------------------------------------------------------//

const static int PaperDirection[2][2][2] = {
{               // duplex type 2 or 4
    { 0, 1 },   // half-fold vertical { not rotated, rotated }
    { 1, 0 },   // half-fold horizontal { not rotated, rotated }
},
{               // duplex type 1 or 3
    { 1, 0 },   // half-fold vertical { not rotated, rotated }
    { 0, 1 },   // half-fold horizontal { not rotated, rotated }
}
};

#define MAX_COLORS  23
const static COLORREF PickColor[MAX_COLORS] = {
    RGB (0,   0,   0  ),
    RGB (255, 255, 255),
    RGB (255, 204, 204),
    RGB (255, 153, 153),
    RGB (255, 51,  153),
    RGB (255, 0,   0  ),
    RGB (204, 0,   0  ),
    RGB (255, 102, 0  ),
    RGB (255, 204, 153),
    RGB (153, 51,  0  ),
    RGB (255, 255, 0  ),
    RGB (255, 204, 0  ),
    RGB (153, 255, 153),
    RGB (0,   255, 0  ),
    RGB (0,   102, 0  ),
    RGB (51,  153, 153),
    RGB (0,   255, 255),
    RGB (102, 153, 204),
    RGB (0,   0,   255),
    RGB (0,   0,   153),
    RGB (255, 153, 255),
    RGB (255, 0,   255),
    RGB (153, 0,   153)
};

//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
static void DeleteRegistryDuplex (const char *pszDevice)
{
    HKEY hKey;
    if (::RegOpenKeyEx (HKEY_LOCAL_MACHINE, DUPLEX_SUBKEY, 0, KEY_WRITE, &hKey)
      == ERROR_SUCCESS)
    {
        ::RegDeleteValue (hKey, pszDevice);
        ::RegCloseKey (hKey);
    }
```

```
}

//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
static bool GetRegistryDuplex (const char *pszDevice, int &nDuplex)
{
    bool bFound = false;

    HKEY hKey;
    if (::RegOpenKeyEx (HKEY_LOCAL_MACHINE, DUPLEX_SUBKEY, 0, KEY_QUERY_VALUE,
      &hKey) == ERROR_SUCCESS)
    {
        char szDuplex[10];
        DWORD dwType;
        DWORD dwSize = sizeof (szDuplex);
        if (::RegQueryValueEx (hKey, pszDevice, 0, &dwType, (BYTE *) szDuplex,
          &dwSize) == ERROR_SUCCESS)
        {
            if (dwType == REG_SZ)
            {
                nDuplex = atoi (szDuplex);
                bFound = true;
            }
        }
        ::RegCloseKey (hKey);
    }

    return (bFound);
}

//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
static void SetRegistryDuplex (const char *pszDevice, int nDuplex)
{
    HKEY hKey;
    DWORD dw;
    if (::RegCreateKeyEx (HKEY_LOCAL_MACHINE, DUPLEX_SUBKEY, 0, REG_NONE,
      REG_OPTION_NON_VOLATILE, KEY_WRITE, NULL, &hKey, &dw) == ERROR_SUCCESS)
    {
        char szDuplex[10];
        _itoa (nDuplex, szDuplex, 10);
        ::RegSetValueEx (hKey, pszDevice, 0, REG_SZ, (BYTE *) szDuplex,
          lstrlen (szDuplex) + 1);
        ::RegCloseKey (hKey);
    }
}

//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
static UINT APIENTRY PrintHookProc (HWND hWnd, UINT msg, UINT wParam,
  LONG lParam)
{
    static bool *pbSingleFold = NULL;

    switch (msg)
    {
        case WM_INITDIALOG:
        {
            PRINTDLG *ppd = (PRINTDLG *) lParam;
            pbSingleFold = (bool *) ppd->lCustData;
            if (*pbSingleFold)
            {
                ::SendDlgItemMessage (hWnd, IDC_SINGLEFOLD, BM_SETCHECK,
                  BST_CHECKED, 0);
            }
            else
            {
                ::SendDlgItemMessage (hWnd, IDC_QUARTERFOLD, BM_SETCHECK,
                  BST_CHECKED, 0);
            }

            int n = ::SendDlgItemMessage (hWnd, cmb4, CB_GETCURSEL, 0, 0);
            if (n >= 0)
            {
                char szDevice[100];
                ::SendDlgItemMessage (hWnd, cmb4, CB_GETLBTEXT, n,
                  (LPARAM) szDevice);

                int nDuplex;
                ::ShowWindow (::GetDlgItem (hWnd, IDC_DBLSIDE),
                  GetRegistryDuplex (szDevice, nDuplex));
            }
            break;
        }

        case WM_COMMAND:
        {
            switch (LOWORD (wParam))
            {
                case IDOK:
                {
                    if (::SendDlgItemMessage (hWnd, IDC_SINGLEFOLD, BM_GETCHECK,
                      0, 0) == BST_CHECKED)
                    {
                        *pbSingleFold = true;
                    }
                    else
                    {
                        *pbSingleFold = false;
                    }

                    if (::IsWindowVisible (::GetDlgItem (hWnd, IDC_DBLSIDE)) &&
                      ::IsWindowEnabled (::GetDlgItem (hWnd, IDC_DBLSIDE)) &&
                      ::SendDlgItemMessage (hWnd, IDC_DBLSIDE, BM_GETCHECK,
                      0, 0) == BST_CHECKED)
                    {
                        int n = ::SendDlgItemMessage (hWnd, cmb4, CB_GETCURSEL,
                          0, 0);
                        if (n >= 0)
                        {
                            char szDevice[100];
                            ::SendDlgItemMessage (hWnd, cmb4, CB_GETLBTEXT, n,
                              (LPARAM) szDevice);
                            DeleteRegistryDuplex (szDevice);
                        }
                    }
                    break;
                }

                case IDC_QUARTERFOLD:
                {
                    ::EnableWindow (::GetDlgItem (hWnd, IDC_DBLSIDE), false);
                    break;
                }

                case IDC_SINGLEFOLD:
                {
                    ::EnableWindow (::GetDlgItem (hWnd, IDC_DBLSIDE), true);
                    break;
                }

                case cmb4:
                {
                    if (HIWORD (wParam) == CBN_SELCHANGE)
                    {
                        int n = ::SendDlgItemMessage (hWnd, cmb4, CB_GETCURSEL,
                          0, 0);
                        if (n >= 0)
                        {
                            char szDevice[100];

                            ::SendDlgItemMessage (hWnd, cmb4, CB_GETLBTEXT, n,
                              (LPARAM) szDevice);

                            int nDuplex;
                            ::ShowWindow (::GetDlgItem (hWnd, IDC_DBLSIDE),
                              GetRegistryDuplex (szDevice, nDuplex));
                        }
                    }
                    break;
                }

                default:
                    break;
            }
        }

        default:
            break;
    }

    return (FALSE);
}

//-------------------------------------------------------------------------------//
//-------------------------------------------------------------------------------//
CCtlPanel::CCtlPanel (CCtp *pMainWnd)
{
    m_pMainWnd = pMainWnd;
    m_pAGDoc = NULL;
    m_hDevMode = NULL;
    m_hDevNames = NULL;
    m_nFontHeight = 0;
    m_hDlg = NULL;
}

//-------------------------------------------------------------------------------//
//-------------------------------------------------------------------------------//
CCtlPanel::~CCtlPanel ()
{
    if (m_hDevMode)
        ::GlobalFree (m_hDevMode);
    if (m_hDevNames)
        ::GlobalFree (m_hDevNames);
    if (m_hDlg)
        ::GlobalFree (m_hDlg);
}

#pragma pack(push, 1)

typedef struct
{
    WORD dlgVer;
    WORD signature;
    DWORD helpID;
    DWORD exStyle;
    DWORD style;
    WORD cDlgItems;
    short x;
    short y;
    short cx;
    short cy;
    short menu;
    short windowClass;
    short title;
    short pointsize;
} DLGTEMPLATEEX;

#pragma pack(pop)


HWND CCtlPanel::Create (HWND hWndParent)
{
    _ASSERTE (m_hWnd == NULL);
    _Module.AddCreateWndData (&m_thunk.cd, (CDialogImplBase *) this);

    HDC hDC = ::GetDC (hWndParent);
    int nLogPixelsY = ::GetDeviceCaps (hDC, LOGPIXELSY);
    ::ReleaseDC (hWndParent, hDC);

    HWND hWnd = NULL;
    if (nLogPixelsY > 96)
    {
        HINSTANCE hInst = _Module.GetResourceInstance ();
        HRSRC hRsrc = ::FindResource (hInst, MAKEINTRESOURCE (CCtlPanel::IDD),
          RT_DIALOG);
        if (hRsrc)
        {
            HGLOBAL hTemplate = ::LoadResource (hInst, hRsrc);
            DLGTEMPLATEEX *pTemplate = (DLGTEMPLATEEX*) ::LockResource (hTemplate);

            int nSize = ::SizeofResource (hInst, hRsrc);
            m_hDlg = ::GlobalAlloc (GPTR, nSize);
            if (m_hDlg)
            {
                DLGTEMPLATEEX *pNew = (DLGTEMPLATEEX *) ::GlobalLock (m_hDlg);
                memcpy ((BYTE *) pNew, pTemplate, nSize);

                pNew->pointsize = (pNew->pointsize * 96 / nLogPixelsY);

                hWnd = ::CreateDialogIndirectParam (hInst, (DLGTEMPLATE *) pNew,
                  hWndParent, (DLGPROC) CCtlPanel::StartDialogProc, NULL);

                ::GlobalUnlock (m_hDlg);
            }
        }
    }
    else
    {
        hWnd = ::CreateDialogParam (_Module.GetResourceInstance(),
          MAKEINTRESOURCE (CCtlPanel::IDD), hWndParent,
          (DLGPROC) CCtlPanel::StartDialogProc, NULL);
    }

    _ASSERTE (m_hWnd == hWnd);
    return hWnd;
}

//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
FONTARRAY &CCtlPanel::GetFontArray ()
{
    return (m_pMainWnd->GetFontList ().GetFontArray ());
}

//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
LRESULT CCtlPanel::OnInitDialog (UINT /*uMsg*/, WPARAM /*wParam*/,
                                 LPARAM /*lParam*/, BOOL & /*bHandled*/)
{
    SendDlgItemMessage (IDC_PAGE1, BM_SETCHECK, TRUE);

    for (int i = 0; i < MAX_COLORS; i++)
    {
        SendDlgItemMessage (IDC_COLOR, CB_ADDSTRING, 0, (LPARAM) "");
        SendDlgItemMessage (IDC_COLOR, CB_SETITEMDATA, i, PickColor[i]);
    }
    SendDlgItemMessage (IDC_COLOR, CB_SETCURSEL, (WPARAM) -1, 0);

    for (i = 8; i <= 72; i += 2)
    {
        char szPtSize[10];
        SendDlgItemMessage (IDC_PTSIZE, CB_ADDSTRING, 0,
          (LPARAM) _itoa (i, szPtSize, 10));
    }
    SendDlgItemMessage (IDC_PTSIZE, CB_SETCURSEL, (WPARAM) -1, 0);
    HFONT hFont = (HFONT) GetStockObject (SYSTEM_FONT);
    SendDlgItemMessage (IDC_PTSIZE, WM_SETFONT, (WPARAM) hFont, 0);

    SetFonts ();
    SendDlgItemMessage (IDC_FONT, CB_SETCURSEL, (WPARAM) -1, 0);

    return (1);
}

//------------------------------------------------------------------------//
//------------------------------------------------------------------------//
LRESULT CCtlPanel::OnCtlColorDlg (UINT /*uMsg*/, WPARAM wParam,
                                  LPARAM /*lParam*/, BOOL &bHandled)
{
    bHandled = TRUE;
    SetBkMode ((HDC) wParam, TRANSPARENT);
    return ((LRESULT) GetStockObject (NULL_BRUSH));
}

//------------------------------------------------------------------------//
//------------------------------------------------------------------------//
LRESULT CCtlPanel::OnCtlColorStatic (UINT /*uMsg*/, WPARAM wParam,
  LPARAM /*lParam*/, BOOL &bHandled)
{
    bHandled = TRUE;
    SetBkMode ((HDC) wParam, TRANSPARENT);
    return ((LRESULT) GetStockObject (NULL_BRUSH));
}

//------------------------------------------------------------------------//
//------------------------------------------------------------------------//
LRESULT CCtlPanel::OnMeasureItem (UINT /*uMsg*/, WPARAM wParam,
                                  LPARAM lParam, BOOL &bHandled)
{
    UINT idCtl = wParam;
    LPMEASUREITEMSTRUCT lpmis = (LPMEASUREITEMSTRUCT) lParam;

    if (idCtl == IDC_FONT)
        m_nFontHeight = lpmis->itemHeight;

    bHandled = TRUE;
    return (TRUE);
}

//------------------------------------------------------------------------//
//------------------------------------------------------------------------//
LRESULT CCtlPanel::OnDrawItem (UINT /*uMsg*/, WPARAM wParam,
                               LPARAM lParam, BOOL &bHandled)
{
    UINT idCtl = wParam;
    LPDRAWITEMSTRUCT lpdis = (LPDRAWITEMSTRUCT) lParam;

    if (idCtl == IDC_COLOR)
    {
        CAGDC dc (lpdis->hDC);
        HBRUSH hbr = (HBRUSH) GetStockObject (WHITE_BRUSH);
        FillRect (lpdis->hDC, &lpdis->rcItem, hbr);

        if ((int) lpdis->itemID != -1)
        {
            COLORREF clr = (COLORREF) ::SendMessage (lpdis->hwndItem,
              CB_GETITEMDATA, lpdis->itemID, 0);

            RECT rect = lpdis->rcItem;
            InflateRect (&rect, -4, -2);
            hbr = CreateSolidBrush (clr | PALETTERGB_FLAG);
            FillRect (lpdis->hDC, &rect, hbr);
            DeleteObject (hbr);
            hbr = (HBRUSH) GetStockObject (BLACK_BRUSH);
            FrameRect (lpdis->hDC, &rect, hbr);
        }

        if (lpdis->itemState & ODS_FOCUS || lpdis->itemState & ODS_SELECTED)
        {
            hbr = (HBRUSH) GetStockObject (BLACK_BRUSH);
            FrameRect (lpdis->hDC, &lpdis->rcItem, hbr);
        }
    }
    else if (idCtl == IDC_FONT)
    {
        if ((int) lpdis->itemID != -1)
        {
            int nFont = ::SendMessage (lpdis->hwndItem, CB_GETITEMDATA, lpdis->itemID, 0);

            FONTARRAY &FontArray = GetFontArray ();
            LOGFONT NewFont = FontArray[nFont].lf;
            NewFont.lfHeight = m_nFontHeight;
            NewFont.lfWidth = 0;
            if (NewFont.lfCharSet == SYMBOL_CHARSET)
            {
                lstrcpy (NewFont.lfFaceName, "Arial");
                NewFont.lfCharSet = ANSI_CHARSET;
                NewFont.lfPitchAndFamily = FF_SWISS;
            }

            SaveDC (lpdis->hDC);
            SetTextAlign (lpdis->hDC, TA_LEFT | TA_TOP | TA_NOUPDATECP);

            if (lpdis->itemState & ODS_SELECTED)
                SetTextColor (lpdis->hDC, GetSysColor (COLOR_HIGHLIGHTTEXT));
            else
                SetTextColor (lpdis->hDC, GetSysColor (COLOR_WINDOWTEXT));

            if (lpdis->itemState & ODS_SELECTED)
                SetBkColor (lpdis->hDC, GetSysColor (COLOR_HIGHLIGHT));
            else
                SetBkColor (lpdis->hDC, GetSysColor (COLOR_WINDOW));

            HFONT hFont = CreateFontIndirect (&NewFont);
            HFONT hOldFont = (HFONT) SelectObject (lpdis->hDC, hFont);
            ExtTextOut (lpdis->hDC, lpdis->rcItem.left, lpdis->rcItem.top,
                ETO_CLIPPED | ETO_OPAQUE, &lpdis->rcItem,
                FontArray[nFont].szFullName,
                lstrlen (FontArray[nFont].szFullName), NULL);

            if (lpdis->itemState & ODS_FOCUS)
                DrawFocusRect (lpdis->hDC, &lpdis->rcItem);

            SelectObject (lpdis->hDC, hOldFont);
            DeleteObject (hFont);
            RestoreDC (lpdis->hDC, -1);
        }
    }

    bHandled = TRUE;
    return (TRUE);
}

//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
LRESULT CCtlPanel::OnPage (WORD /*wNotifyCode*/, WORD wID, HWND /*hWndCtl*/,
                           BOOL &bHandled)
{
    if (m_pAGDoc)
    {
        m_pAGDoc->SetCurrentPage (m_PageMap[wID - IDC_PAGE1]);
        m_pMainWnd->NewPage ();
    }

    m_pMainWnd->SetFocus ();

    bHandled = TRUE;
    return (TRUE);
}

//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
LRESULT CCtlPanel::OnPrint (WORD /*wNotifyCode*/, WORD /*wID*/, HWND /*hWndCtl*/,
                            BOOL & /*bHandled*/)
{
    if (m_pAGDoc)
    {
        AGDOCTYPE DocType = m_pAGDoc->GetDocType ();
        bool bSingleFold = (DocType == DOC_CARDHV || DocType == DOC_CARDHH);

        PRINTDLG pd;
        memset (&pd, 0, sizeof (pd));
        pd.lStructSize = sizeof (pd);
        pd.hwndOwner = GetParent ();
        pd.hDevMode = m_hDevMode;
        pd.hDevNames = m_hDevNames;
        pd.hInstance = _Module.GetResourceInstance ();
        pd.lCustData = (DWORD) &bSingleFold;
        pd.lpfnPrintHook = PrintHookProc;
        pd.lpPrintTemplateName = MAKEINTRESOURCE (PRINTDLGORD);
        pd.Flags = PD_ENABLEPRINTTEMPLATE | PD_ENABLEPRINTHOOK;

        if (PrintDlg (&pd))
        {
            DEVNAMES *pDevNames = (DEVNAMES *) GlobalLock (pd.hDevNames);
            DEVMODE *pDevMode = (DEVMODE *) GlobalLock (pd.hDevMode);
            char *pszDriver = ((char *) pDevNames) + pDevNames->wDriverOffset;
            char *pszDevice = ((char *) pDevNames) + pDevNames->wDeviceOffset;
            char *pszOutput = ((char *) pDevNames) + pDevNames->wOutputOffset;

            int nDuplex;
            if (! GetRegistryDuplex (pszDevice, nDuplex))
                nDuplex = -1;

            if (bSingleFold && nDuplex == -1)
            {
                CDblSideIntro   Intro;
                CDblSideStep1   Step1;
                CDblSideStep2   Step2;
                CDblSideEnd     End;

                PROPSHEETPAGE *pPropPages = new PROPSHEETPAGE[4];
                pPropPages[0] = Intro.m_psp;
                pPropPages[1] = Step1.m_psp;
                pPropPages[2] = Step2.m_psp;
                pPropPages[3] = End.m_psp;

                Intro.m_pszDriver = Step1.m_pszDriver = pszDriver;
                Intro.m_pszDevice = Step1.m_pszDevice = pszDevice;
                Intro.m_pszOutput = Step1.m_pszOutput = pszOutput;
                Intro.m_pDevMode = Step1.m_pDevMode = pDevMode;

                PROPSHEETHEADER psh;
                psh.dwSize = sizeof (PROPSHEETHEADER);
                psh.dwFlags = PSH_WIZARD | PSH_PROPSHEETPAGE;
                psh.hwndParent = GetParent ();
                psh.hInstance = NULL;
                psh.hIcon = NULL;
                psh.pszCaption = NULL;
                psh.nPages = 4;
                psh.nStartPage = 0;
                psh.ppsp = pPropPages;
                psh.pfnCallback = NULL;

                ::PropertySheet (&psh);
                if (End.IsFinished ())
                {
                    nDuplex = Step2.GetSelected ();
                    SetRegistryDuplex (pszDevice, nDuplex);
                }

                delete [] pPropPages;
            }

            if (! (bSingleFold && nDuplex == -1))
            {
                CWaitDlg WaitDlg;
                WaitDlg.Create (GetParent());
                WaitDlg.UpdateWindow();

                MSG msg;
                while (PeekMessage(&msg, NULL, 0, 0, PM_REMOVE))
                {
                    TranslateMessage(&msg);
                    DispatchMessage(&msg);
                }

                bool bRotated;
                int nCopies = pd.nCopies;
                while (nCopies-- > 0)
                {
                    if (bSingleFold)
                    {
                        m_pAGDoc->PrintCardSingle (PRINT_OUTSIDE, pszDriver,
                          pszDevice, pszOutput, pDevMode, bRotated);
                    }
                    else
                    {
                        m_pAGDoc->PrintCardQuarter (pszDriver, pszDevice,
                          pszOutput, pDevMode);
                    }
                }

                WaitDlg.DestroyWindow();

                if (bSingleFold)
                {
                    char szFace[10];
                    if (nDuplex > 2)
                        lstrcpy (szFace, "UP");
                    else
                        lstrcpy (szFace, "DOWN");

                    DWORD dwOrientation = ::DeviceCapabilities (pszDevice,
                      pszOutput, DC_ORIENTATION, NULL, pDevMode);

                    bool bHorz = (DocType == DOC_CARDHH || DocType == DOC_CARDFH);
                    if (!bHorz && dwOrientation == 270)
                        bRotated = !bRotated;

                    char szDirection[15];
                    int nDirection = PaperDirection[nDuplex % 2][bHorz][bRotated];
                    if (nDirection == 1)
                        lstrcpy (szDirection, "AWAY FROM");
                    else
                        lstrcpy (szDirection, "TOWARD");

                    char szMsg[256];
                    wsprintf (szMsg, "To print the inside of your card, reinsert the page with the p
rinted side %s\nand the front panel of the card %s the printer.\n\n\nClick OK when you are ready to
print the inside.",
                      szFace, szDirection);

                    if (MessageBox (szMsg, "Print inside", MB_OKCANCEL) == IDOK)
                    {
                        WaitDlg.Create (GetParent());
                        WaitDlg.UpdateWindow();

                        MSG msg;
                        while (PeekMessage(&msg, NULL, 0, 0, PM_REMOVE))
                        {
                            TranslateMessage(&msg);
                            DispatchMessage(&msg);
                        }

                        nCopies = pd.nCopies;
                        while (nCopies-- > 0)
                        {
                            m_pAGDoc->PrintCardSingle (PRINT_INSIDE, pszDriver,
                              pszDevice, pszOutput, pDevMode, bRotated);
                        }
                        WaitDlg.DestroyWindow();
                    }
                }
            }

            GlobalUnlock (pd.hDevNames);
            GlobalUnlock (pd.hDevMode);
        }

        m_hDevMode = pd.hDevMode;
        m_hDevNames = pd.hDevNames;

        m_pMainWnd->SetFocus ();
    }

    return (TRUE);
}

//----------------------------------------------------------------------------//
//----------------------------------------------------------------------------//
void CCtlPanel::SetDoc (CAGDoc *pAGDoc)
{
    m_pAGDoc = pAGDoc;

    AGDOCTYPE DocType = m_pAGDoc->GetDocType ();
    bool bCardType = (DocType == DOC_CARDHV || DocType == DOC_CARDHH ||
      DocType == DOC_CARDFV || DocType == DOC_CARDFH);

    int nPages = m_pAGDoc->GetNumPages ();
    for (int nPage = 1, nMap = 0; nPage <= nPages; nPage++)
    {
        CAGPage *pPage = m_pAGDoc->GetPage (nPage);

        if ((! pPage->IsEmpty ()) || (bCardType && nPage == 4))
        {
            const char *pszPageName = pPage->GetPageName ();
            SetDlgItemText (IDC_PAGE1 + nMap, pszPageName);
            ::ShowWindow (GetDlgItem (IDC_PAGE1 + nMap), SW_SHOW);
            m_PageMap[nMap++] = nPage;
        }
    }

    while (nMap < 4)
    {
        ::ShowWindow (GetDlgItem (IDC_PAGE1 + nMap), SW_HIDE);
        m_PageMap[nMap++] = 0;
    }
}

//----------------------------------------------------------------------------//
//----------------------------------------------------------------------------//
void CCtlPanel::UpdateControls (const CAGText *pText)
{
    scTypeSpecList  tsList;
    int             nNumItems = 0;

    if (pText)
    {
        pText->GetSelTSList (tsList);
        nNumItems = tsList.NumItems ();
    }

    if (nNumItems == 0)
    {
        SendDlgItemMessage (IDC_FONT, CB_SETCURSEL, (WPARAM) -1, 0);
        SendDlgItemMessage (IDC_PTSIZE, CB_SETCURSEL, (WPARAM) -1, 0);
        SendDlgItemMessage (IDC_COLOR, CB_SETCURSEL, (WPARAM) -1, 0);
        SendDlgItemMessage (IDC_LEFT, BM_SETCHECK, FALSE);
        SendDlgItemMessage (IDC_CENTER, BM_SETCHECK, FALSE);
        SendDlgItemMessage (IDC_RIGHT, BM_SETCHECK, FALSE);

        return;
    }

    CAGSpec     *pAGSpec = (CAGSpec *) (tsList[0].ptr ());
    LOGFONT     SpecFont = pAGSpec->m_Font;
    int         nSpecPtSize = abs (pAGSpec->m_Font.lfHeight) * 72 / APP_RESOLUTION;
    COLORREF    SpecColor = pAGSpec->m_Color;
    bool        bFont = true;
    bool        bPtSize = true;
    bool        bColor = true;

    if (nNumItems > 1)
    {
        for (int i = 1; i < nNumItems; i++)
        {
            pAGSpec = (CAGSpec *) (tsList[i].ptr ());
            if (lstrcmp (SpecFont.lfFaceName, pAGSpec->m_Font.lfFaceName) != 0
              || SpecFont.lfWeight != pAGSpec->m_Font.lfWeight
              || (SpecFont.lfItalic != 0) != (pAGSpec->m_Font.lfItalic != 0))
            {
                bFont = false;
            }

            int nPtSize = abs (pAGSpec->m_Font.lfHeight) * 72 / APP_RESOLUTION;
            if (nSpecPtSize != nPtSize)
                bPtSize = false;

            if (SpecColor != pAGSpec->m_Color)
                bColor = false;
        }
    }

    if (bFont)
    {
        FONTARRAY &FontArray = GetFontArray ();
        int nFonts = FontArray.size ();
        for (int i = 0; i < nFonts; i++)
        {
            if (lstrcmp (SpecFont.lfFaceName, FontArray[i].lf.lfFaceName) == 0
              && SpecFont.lfWeight == FontArray[i].lf.lfWeight
              && (SpecFont.lfItalic != 0) == (FontArray[i].lf.lfItalic != 0))
            {
                break;
            }
        }

        if (i >= nFonts)
            SendDlgItemMessage (IDC_FONT, CB_SETCURSEL, (WPARAM) -1, 0);
        else
        {
            int index = SendDlgItemMessage (IDC_FONT, CB_FINDSTRINGEXACT,
              (WPARAM) -1, (LPARAM) FontArray[i].szFullName);
            SendDlgItemMessage (IDC_FONT, CB_SETCURSEL, index, 0);
        }
    }
    else
        SendDlgItemMessage (IDC_FONT, CB_SETCURSEL, (WPARAM) -1, 0);

    if (bPtSize)
    {
        char szPtSize[10];
        int nPtSize = abs (SpecFont.lfHeight) * 72 / APP_RESOLUTION;
        int index = SendDlgItemMessage (IDC_PTSIZE, CB_FINDSTRINGEXACT,
          (WPARAM) -1, (LPARAM) _itoa (nPtSize, szPtSize, 10));
        if (index == CB_ERR)
        {
            int nItems = SendDlgItemMessage (IDC_PTSIZE, CB_GETCOUNT, 0, 0);
            for (int i = 0; i < nItems; i++)
            {
                char szTemp[20];
                SendDlgItemMessage (IDC_PTSIZE, CB_GETLBTEXT, i,
                  (LPARAM) szTemp);
                if (nPtSize < atoi (szTemp))
                {
                    index = SendDlgItemMessage (IDC_PTSIZE, CB_INSERTSTRING, i,
                      (LPARAM) szPtSize);
                    break;
                }
            }
            if (i >= nItems)
            {
                index = SendDlgItemMessage (IDC_PTSIZE, CB_ADDSTRING, 0,
                  (LPARAM) szPtSize);
            }
        }
        SendDlgItemMessage (IDC_PTSIZE, CB_SETCURSEL, index, 0);
    }
    else
        SendDlgItemMessage (IDC_PTSIZE, CB_SETCURSEL, (WPARAM) -1, 0);

    if (bColor)
    {
        int nItems = SendDlgItemMessage (IDC_COLOR, CB_GETCOUNT, 0, 0);
        for (int i = 0; i < nItems; i++)
        {
            COLORREF Color = (COLORREF) SendDlgItemMessage (IDC_COLOR,
              CB_GETITEMDATA, i, 0);
            if (Color == SpecColor)
                break;
        }

        if (i >= nItems)
        {
            SendDlgItemMessage (IDC_COLOR, CB_INSERTSTRING, 0, (LPARAM) "");
            SendDlgItemMessage (IDC_COLOR, CB_SETITEMDATA, 0, SpecColor);
            i = 0;
        }

        SendDlgItemMessage (IDC_COLOR, CB_SETCURSEL, i, 0);
    }
    else
        SendDlgItemMessage (IDC_COLOR, CB_SETCURSEL, (WPARAM) -1, 0);

    scTypeSpecList tsListPara;
    pText->GetSelParaTSList (tsListPara);
    nNumItems = tsListPara.NumItems ();

    pAGSpec = (CAGSpec *) (tsListPara[0].ptr ());
    eTSJust SpecHorzJust = pAGSpec->m_HorzJust;
    bool    bHorzJust = true;

    if (nNumItems > 1)
    {
        for (int i = 1; i < nNumItems; i++)
        {
            pAGSpec = (CAGSpec *) (tsListPara[i].ptr ());
            if (SpecHorzJust != pAGSpec->m_HorzJust)
                bHorzJust = false;
        }
    }

    if (bHorzJust)
    {
        switch (SpecHorzJust)
        {
            case eRagCentered:
                SendDlgItemMessage (IDC_CENTER, BM_SETCHECK, TRUE);
                SendDlgItemMessage (IDC_LEFT, BM_SETCHECK, FALSE);
                SendDlgItemMessage (IDC_RIGHT, BM_SETCHECK, FALSE);
                break;

            case eRagLeft:
                SendDlgItemMessage (IDC_RIGHT, BM_SETCHECK, TRUE);
                SendDlgItemMessage (IDC_LEFT, BM_SETCHECK, FALSE);
                SendDlgItemMessage (IDC_CENTER, BM_SETCHECK, FALSE);
                break;
            case eRagRight:
            default:
                SendDlgItemMessage (IDC_LEFT, BM_SETCHECK, TRUE);
                SendDlgItemMessage (IDC_CENTER, BM_SETCHECK, FALSE);
                SendDlgItemMessage (IDC_RIGHT, BM_SETCHECK, FALSE);
                break;
        }
    }
    else
    {
        SendDlgItemMessage (IDC_LEFT, BM_SETCHECK, FALSE);
        SendDlgItemMessage (IDC_CENTER, BM_SETCHECK, FALSE);
        SendDlgItemMessage (IDC_RIGHT, BM_SETCHECK, FALSE);
    }
}

//----------------------------------------------------------------------------//
//----------------------------------------------------------------------------//
LRESULT CCtlPanel::OnHorzJust (WORD /*wNotifyCode*/, WORD wID, HWND /*hWndCtl*/,
  BOOL &bHandled)
{
    CAGText *pText = m_pMainWnd->GetText ();
    if (pText)
    {
        switch (wID)
        {
            case IDC_LEFT:
                pText->SetHorzJust (eRagRight);
                break;

            case IDC_CENTER:
                pText->SetHorzJust (eRagCentered);
                break;

            case IDC_RIGHT:
                pText->SetHorzJust (eRagLeft);
                break;
        }
    }

    m_pMainWnd->SetFocus ();

    bHandled = TRUE;
    return (TRUE);
}

//----------------------------------------------------------------------------//
//----------------------------------------------------------------------------//
LRESULT CCtlPanel::OnFont (WORD wNotifyCode, WORD wID, HWND hWndCtl,
BOOL &bHandled)
{
    if (wNotifyCode != CBN_SELCHANGE)
    {
        if (wNotifyCode == CBN_CLOSEUP)
            m_pMainWnd->SetFocus ();

        bHandled = TRUE;
        return (TRUE);
    }

    int nItem = SendDlgItemMessage (IDC_FONT, CB_GETCURSEL, 0, 0);
    if (nItem >= 0)
    {
        CAGText *pText = m_pMainWnd->GetText ();
        if (pText)
        {
            int nFont = SendDlgItemMessage (IDC_FONT, CB_GETITEMDATA, nItem, 0);
            FONTARRAY &FontArray = GetFontArray ();
            pText->SetTypeface (FontArray[nFont].lf);
        }
    }
    m_pMainWnd->SetFocus ();
    bHandled = TRUE;
    return (TRUE);
}

//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
LRESULT CCtlPanel::OnPtSize (WORD wNotifyCode, WORD wID, HWND hWndCtl,
  BOOL &bHandled)
{
    if (wNotifyCode != CBN_SELCHANGE)
    {
        if (wNotifyCode == CBN_CLOSEUP)
            m_pMainWnd->SetFocus ();

        bHandled = TRUE;
        return (TRUE);
    }

    int nItem = SendDlgItemMessage (IDC_PTSIZE, CB_GETCURSEL, 0, 0);
    if (nItem >= 0)
    {
        CAGText *pText = m_pMainWnd->GetText ();
        if (pText)
        {
            char szPtSize[20];
            SendDlgItemMessage (IDC_PTSIZE, CB_GETLBTEXT, nItem,
              (LPARAM) szPtSize);
            pText->SetPtSize (atoi (szPtSize));
        }
    }
    m_pMainWnd->SetFocus ();

    bHandled = TRUE;
    return (TRUE);
}

//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
LRESULT CCtlPanel::OnColor (WORD wNotifyCode, WORD wID, HWND hWndCtl,
  BOOL &bHandled)
{
    if (wNotifyCode != CBN_SELCHANGE)
    {
        if (wNotifyCode == CBN_CLOSEUP)
            m_pMainWnd->SetFocus ();

        bHandled = TRUE;
        return (TRUE);
    }

    int nItem = SendDlgItemMessage (IDC_COLOR, CB_GETCURSEL, 0, 0);
    if (nItem >= 0)
    {
        CAGText *pText = m_pMainWnd->GetText ();
        if (pText)
        {
            COLORREF Color = (COLORREF) SendDlgItemMessage (IDC_COLOR,
              CB_GETITEMDATA, nItem, 0);
            pText->SetColor (Color);
        }
    }
    m_pMainWnd->SetFocus ();

    bHandled = TRUE;
    return (TRUE);
}

//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
void CCtlPanel::SetFonts ()
{
    SendDlgItemMessage (IDC_FONT, CB_RESETCONTENT, 0, 0);
    FONTARRAY &FontArray = GetFontArray ();
    int nFonts = FontArray.size ();
    for (int i = 0; i < nFonts; i++)
    {
        if (SendDlgItemMessage (IDC_FONT, CB_FINDSTRINGEXACT,
          (WPARAM) -1, (LPARAM) FontArray[i].szFullName) == CB_ERR)
        {
            int index = SendDlgItemMessage (IDC_FONT, CB_ADDSTRING, 0,
              (LPARAM) FontArray[i].szFullName);
            SendDlgItemMessage (IDC_FONT, CB_SETITEMDATA, index, i);
        }
    }
}
```

```
#ifndef __BSC2_H_
#define __BSC2_H_

template <class T>
class CBindStatusCallback2 : public CBindStatusCallback<T>
{
public:
    STDMETHOD(OnProgress)(ULONG ulProgress, ULONG ulProgressMax, ULONG ulStatusCode, LPCWSTR szStatu
sText)
    {
        ATLTRACE(_T("CBindStatusCallback2::OnProgress"));
        return m_pT->OnProgress ( ulProgress, ulProgressMax, ulStatusCode, szStatusText );
    }

    static HRESULT Download(T* pT, ATL_PDATAAVAILABLE pFunc, BSTR bstrURL, IUnknown* pUnkContainer =
 NULL, BOOL bRelative = FALSE)
    {
        CComObject<CBindStatusCallback2<T> > *pbsc;
        HRESULT hRes = CComObject<CBindStatusCallback2<T> >::CreateInstance(&pbsc);
        if (FAILED(hRes))
            return hRes;
        return pbsc->StartAsyncDownload(pT, pFunc, bstrURL, pUnkContainer, bRelative);
    }
};

#endif //__BSC2_H_
```

```
/* this ALWAYS GENERATED file contains the proxy stub code */


/* File created by MIDL compiler version 5.01.0164 */
/* at Thu Mar 09 10:57:11 2000
 */
/* Compiler settings for AxCtp.idl:
    Oicf (OptLev=i2), W1, Zp8, env=Win32, ms_ext, c_ext
    error checks: allocation ref bounds_check enum stub_data
*/
//@@MIDL_FILE_HEADING(  )

#define USE_STUBLESS_PROXY


/* verify that the <rpcproxy.h> version is high enough to compile this file*/
#ifndef __REDQ_RPCPROXY_H_VERSION__
#define __REQUIRED_RPCPROXY_H_VERSION__ 440
#endif


#include "rpcproxy.h"
#ifndef __RPCPROXY_H_VERSION__
#error this stub requires an updated version of <rpcproxy.h>
#endif // __RPCPROXY_H_VERSION__


#include "AxCtp.h"

#define TYPE_FORMAT_STRING_SIZE   55
#define PROC_FORMAT_STRING_SIZE   113

typedef struct _MIDL_TYPE_FORMAT_STRING
    {
    short          Pad;
    unsigned char  Format[ TYPE_FORMAT_STRING_SIZE ];
    } MIDL_TYPE_FORMAT_STRING;

typedef struct _MIDL_PROC_FORMAT_STRING
    {
    short          Pad;
    unsigned char  Format[ PROC_FORMAT_STRING_SIZE ];
    } MIDL_PROC_FORMAT_STRING;


extern const MIDL_TYPE_FORMAT_STRING __MIDL_TypeFormatString;
extern const MIDL_PROC_FORMAT_STRING __MIDL_ProcFormatString;


/* Object interface: IUnknown, ver. 0.0,
   GUID={0x00000000,0x0000,0x0000,{0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46}} */


/* Object interface: IDispatch, ver. 0.0,
   GUID={0x00020400,0x0000,0x0000,{0xC0,0x00,0x00,0x00,0x00,0x00,0x00,0x46}} */


/* Object interface: ICtp, ver. 0.0,
   GUID={0x38578BFE,0x0ABB,0x11D3,{0x93,0x30,0x00,0x80,0xC6,0xF7,0x96,0xA1}} */


extern const MIDL_STUB_DESC Object_StubDesc;


extern const MIDL_SERVER_INFO ICtp_ServerInfo;

#pragma code_seg(".orpc")
extern const USER_MARSHAL_ROUTINE_QUADRUPLE UserMarshalRoutines[1];

static const MIDL_STUB_DESC Object_StubDesc =
    {
    0,
    NdrOleAllocate,
    NdrOleFree,
    0,
    0,
    0,
    0,
    0,
    __MIDL_TypeFormatString.Format,
    1, /* -error bounds_check flag */
    0x20000, /* Ndr library version */
    0,
    0x50100a4, /* MIDL Version 5.1.164 */
    0,
    UserMarshalRoutines,
    0,  /* notify & notify_flag routine table */
    1,  /* Flags */
    0,  /* Reserved3 */
    0,  /* Reserved4 */
    0   /* Reserved5 */
    };

static const unsigned short ICtp_FormatStringOffsetTable[] =
    {
    (unsigned short) -1,
    (unsigned short) -1,
    (unsigned short) -1,
    (unsigned short) -1,
    0,
    28,
    56,
    84
    };

static const MIDL_SERVER_INFO ICtp_ServerInfo =
    {
    &Object_StubDesc,
    0,
    __MIDL_ProcFormatString.Format,
    &ICtp_FormatStringOffsetTable[-3],
    0,
    0,
    0,
    0
    };

static const MIDL_STUBLESS_PROXY_INFO ICtp_ProxyInfo =
    {
    &Object_StubDesc,
    __MIDL_ProcFormatString.Format,
    &ICtp_FormatStringOffsetTable[-3],
    0,
    0,
    0
    };

CINTERFACE_PROXY_VTABLE(11) _ICtpProxyVtbl =
{
    &ICtp_ProxyInfo,
    &IID_ICtp,
    IUnknown_QueryInterface_Proxy,
    IUnknown_AddRef_Proxy,
    IUnknown_Release_Proxy ,
    0 /* (void *)-1 /* IDispatch::GetTypeInfoCount */ ,
    0 /* (void *)-1 /* IDispatch::GetTypeInfo */ ,
    0 /* (void *)-1 /* IDispatch::GetIDsOfNames */ ,
    0 /* IDispatch_Invoke_Proxy */ ,
    (void *)-1 /* ICtp::put_Fonts */ ,
    (void *)-1 /* ICtp::get_Fonts */ ,
    (void *)-1 /* ICtp::put_Src */ ,
    (void *)-1 /* ICtp::get_Src */
};


static const PRPC_STUB_FUNCTION ICtp_table[] =
{
    STUB_FORWARDING_FUNCTION,
    STUB_FORWARDING_FUNCTION,
    STUB_FORWARDING_FUNCTION,
    STUB_FORWARDING_FUNCTION,
    NdrStubCall2,
    NdrStubCall2,
    NdrStubCall2,
    NdrStubCall2
};

CInterfaceStubVtbl _ICtpStubVtbl =
{
    &IID_ICtp,
    &ICtp_ServerInfo,
    11,
    &ICtp_table[-3],
    CStdStubBuffer_DELEGATING_METHODS
};

#pragma data_seg(".rdata")

static const USER_MARSHAL_ROUTINE_QUADRUPLE UserMarshalRoutines[1] =
        {

            {
            BSTR_UserSize
            ,BSTR_UserMarshal
            ,BSTR_UserUnmarshal
            ,BSTR_UserFree
            }

        };


#if !defined(__RPC_WIN32__)
#error  Invalid build platform for this stub.
#endif

#if !(TARGET_IS_NT40_OR_LATER)
#error You need a Windows NT 4.0 or later to run this stub because it uses these features:
#error   -Oif or -Oicf, [wire_marshal] or [user_marshal] attribute, more than 32 methods in the inte
rface.
#error However, your C/C++ compilation flags indicate you intend to run this app on earlier systems.
#error This app will die there with the RPC_X_WRONG_STUB_VERSION error.
#endif


static const MIDL_PROC_FORMAT_STRING __MIDL_ProcFormatString =
    {
        0,
        {

    /* Procedure put_Fonts */

            0x33,       /* FC_AUTO_HANDLE */
            0x6c,       /* Old Flags:  object, Oi2 */
/*  2 */    NdrFcLong( 0x0 ),   /* 0 */
/*  6 */    NdrFcShort( 0x7 ),  /* 7 */
#ifndef _ALPHA_
/*  8 */    NdrFcShort( 0xc ),  /* x86, MIPS, PPC Stack size/offset = 12 */
#else
            NdrFcShort( 0x18 ), /* Alpha Stack size/offset = 24 */
#endif
/* 10 */    NdrFcShort( 0x0 ),  /* 0 */
/* 12 */    NdrFcShort( 0x8 ),  /* 8 */
/* 14 */    0x6,        /* Oi2 Flags:  clt must size, has return, */
            0x2,        /* 2 */

    /* Parameter strFonts */

/* 16 */    NdrFcShort( 0x8b ), /* Flags:  must size, must free, in, by val, */
#ifndef _ALPHA_
```

```
/* 18 */    NdrFcShort( 0x4 ),  /* x86, MIPS, PPC Stack size/offset = 4 */
#else
            NdrFcShort( 0x8 ),  /* Alpha Stack size/offset = 8 */
#endif
/* 20 */    NdrFcShort( 0x1a ), /* Type Offset=26 */

    /* Return value */

/* 22 */    NdrFcShort( 0x70 ), /* Flags:  out, return, base type, */
#ifndef _ALPHA_
/* 24 */    NdrFcShort( 0x8 ),  /* x86, MIPS, PPC Stack size/offset = 8 */
#else
            NdrFcShort( 0x10 ), /* Alpha Stack size/offset = 16 */
#endif
/* 26 */    0x8,        /* FC_LONG */
            0x0,        /* 0 */

    /* Procedure get_Fonts */

/* 28 */    0x33,       /* FC_AUTO_HANDLE */
            0x6c,       /* Old Flags:  object, Oi2 */
/* 30 */    NdrFcLong( 0x0 ),   /* 0 */
/* 34 */    NdrFcShort( 0x8 ),  /* 8 */
#ifndef _ALPHA_
/* 36 */    NdrFcShort( 0xc ),  /* x86, MIPS, PPC Stack size/offset = 12 */
#else
            NdrFcShort( 0x18 ), /* Alpha Stack size/offset = 24 */
#endif
/* 38 */    NdrFcShort( 0x0 ),  /* 0 */
/* 40 */    NdrFcShort( 0x8 ),  /* 8 */
/* 42 */    0x5,        /* Oi2 Flags:  srv must size, has return, */
            0x2,        /* 2 */

    /* Parameter pstrFonts */

/* 44 */    NdrFcShort( 0x2113 ),   /* Flags:  must size, must free, out, simple ref, srv alloc size
=8 */
#ifndef _ALPHA_
/* 46 */    NdrFcShort( 0x4 ),  /* x86, MIPS, PPC Stack size/offset = 4 */
#else
            NdrFcShort( 0x8 ),  /* Alpha Stack size/offset = 8 */
#endif
/* 48 */    NdrFcShort( 0x2c ), /* Type Offset=44 */

    /* Return value */

/* 50 */    NdrFcShort( 0x70 ), /* Flags:  out, return, base type, */
#ifndef _ALPHA_
/* 52 */    NdrFcShort( 0x8 ),  /* x86, MIPS, PPC Stack size/offset = 8 */
#else
            NdrFcShort( 0x10 ), /* Alpha Stack size/offset = 16 */
#endif
/* 54 */    0x8,        /* FC_LONG */
            0x0,        /* 0 */

    /* Procedure put_Src */

/* 56 */    0x33,       /* FC_AUTO_HANDLE */
            0x6c,       /* Old Flags:  object, Oi2 */
/* 58 */    NdrFcLong( 0x0 ),   /* 0 */
/* 62 */    NdrFcShort( 0x9 ),  /* 9 */
#ifndef _ALPHA_
/* 64 */    NdrFcShort( 0xc ),  /* x86, MIPS, PPC Stack size/offset = 12 */
#else
            NdrFcShort( 0x18 ), /* Alpha Stack size/offset = 24 */
#endif
/* 66 */    NdrFcShort( 0x0 ),  /* 0 */
/* 68 */    NdrFcShort( 0x8 ),  /* 8 */
/* 70 */    0x6,        /* Oi2 Flags:  clt must size, has return, */
            0x2,        /* 2 */

    /* Parameter strSrc */
/* 72 */    NdrFcShort( 0x8b ), /* Flags:  must size, must free, in, by val, */
#ifndef _ALPHA_
/* 74 */    NdrFcShort( 0x4 ),  /* x86, MIPS, PPC Stack size/offset = 4 */
#else
            NdrFcShort( 0x8 ),  /* Alpha Stack size/offset = 8 */
#endif
/* 76 */    NdrFcShort( 0x1a ), /* Type Offset=26 */

    /* Return value */

/* 78 */    NdrFcShort( 0x70 ), /* Flags:  out, return, base type, */
#ifndef _ALPHA_
/* 80 */    NdrFcShort( 0x8 ),  /* x86, MIPS, PPC Stack size/offset = 8 */
#else
            NdrFcShort( 0x10 ), /* Alpha Stack size/offset = 16 */
#endif
/* 82 */    0x8,        /* FC_LONG */
            0x0,        /* 0 */

    /* Procedure get_Src */

/* 84 */    0x33,       /* FC_AUTO_HANDLE */
            0x6c,       /* Old Flags:  object, Oi2 */
/* 86 */    NdrFcLong( 0x0 ),   /* 0 */
/* 90 */    NdrFcShort( 0xa ),  /* 10 */
#ifndef _ALPHA_
/* 92 */    NdrFcShort( 0xc ),  /* x86, MIPS, PPC Stack size/offset = 12 */
#else
            NdrFcShort( 0x18 ), /* Alpha Stack size/offset = 24 */
#endif
/* 94 */    NdrFcShort( 0x0 ),  /* 0 */
/* 96 */    NdrFcShort( 0x8 ),  /* 8 */
/* 98 */    0x5,        /* Oi2 Flags:  srv must size, has return, */
            0x2,        /* 2 */

    /* Parameter pstrSrc */

/* 100 */   NdrFcShort( 0x2113 ),   /* Flags:  must size, must free, out, simple ref, srv alloc size
=8 */
#ifndef _ALPHA_
/* 102 */   NdrFcShort( 0x4 ),  /* x86, MIPS, PPC Stack size/offset = 4 */
#else
            NdrFcShort( 0x8 ),  /* Alpha Stack size/offset = 8 */
#endif
/* 104 */   NdrFcShort( 0x2c ), /* Type Offset=44 */

    /* Return value */

/* 106 */   NdrFcShort( 0x70 ), /* Flags:  out, return, base type, */
#ifndef _ALPHA_
/* 108 */   NdrFcShort( 0x8 ),  /* x86, MIPS, PPC Stack size/offset = 8 */
#else
            NdrFcShort( 0x10 ), /* Alpha Stack size/offset = 16 */
#endif
/* 110 */   0x8,        /* FC_LONG */
            0x0,        /* 0 */

            0x0
        }
    };

static const MIDL_TYPE_FORMAT_STRING __MIDL_TypeFormatString =
    {
        0,
        {
            NdrFcShort( 0x0 ),  /* 0 */
/*  2 */
            0x12, 0x0,  /* FC_UP */
/*  4 */    NdrFcShort( 0xc ),  /* Offset= 12 (16) */
/*  6 */
            0x1b,       /* FC_CARRAY */
            0x1,        /* 1 */
/*  8 */    NdrFcShort( 0x2 ),  /* 2 */
/* 10 */    0x9,         /* Corr desc: FC_ULONG */
            0x0,         /*  */
/* 12 */    NdrFcShort( 0xfffc ),   /* -4 */
/* 14 */    0x6,         /* FC_SHORT */
            0x5b,        /* FC_END */
/* 16 */
            0x17,        /* FC_CSTRUCT */
            0x3,         /* 3 */
/* 18 */    NdrFcShort( 0x8 ),  /* 8 */
/* 20 */    NdrFcShort( 0xfffffff2 ),   /* Offset= -14 (6) */
/* 22 */    0x8,         /* FC_LONG */
            0x8,         /* FC_LONG */
/* 24 */    0x5c,        /* FC_PAD */
            0x5b,        /* FC_END */
/* 26 */    0xb4,        /* FC_USER_MARSHAL */
            0x83,        /* 131 */
/* 28 */    NdrFcShort( 0x0 ),  /* 0 */
/* 30 */    NdrFcShort( 0x4 ),  /* 4 */
/* 32 */    NdrFcShort( 0x0 ),  /* 0 */
/* 34 */    NdrFcShort( 0xffffffe0 ),   /* Offset= -32 (2) */
/* 36 */
            0x11, 0x4,  /* FC_RP [alloced_on_stack] */
/* 38 */    NdrFcShort( 0x6 ),  /* Offset= 6 (44) */
/* 40 */
            0x13, 0x0,  /* FC_OP */
/* 42 */    NdrFcShort( 0xffffffe6 ),   /* Offset= -26 (16) */
/* 44 */    0xb4,        /* FC_USER_MARSHAL */
            0x83,        /* 131 */
/* 46 */    NdrFcShort( 0x0 ),  /* 0 */
/* 48 */    NdrFcShort( 0x4 ),  /* 4 */
/* 50 */    NdrFcShort( 0x0 ),  /* 0 */
/* 52 */    NdrFcShort( 0xfffffff4 ),   /* Offset= -12 (40) */

            0x0
        }
    };

const CInterfaceProxyVtbl * _AxCtp_ProxyVtblList[] =
{
    ( CInterfaceProxyVtbl *) &_ICtpProxyVtbl,
    0
};

const CInterfaceStubVtbl * _AxCtp_StubVtblList[] =
{
    ( CInterfaceStubVtbl *) &_ICtpStubVtbl,
    0
};

PCInterfaceName const _AxCtp_InterfaceNamesList[] =
{
    "ICtp",
    0
};

const IID *  _AxCtp_BaseIIDList[] =
{
    &IID_IDispatch,
    0
};


#define _AxCtp_CHECK_IID(n) IID_GENERIC_CHECK_IID( _AxCtp, pIID, n)

int __stdcall _AxCtp_IID_Lookup( const IID * pIID, int * pIndex )
{

    if(!_AxCtp_CHECK_IID(0))
        {
        *pIndex = 0;
        return 1;
        }
    return 0;
}

const ExtendedProxyFileInfo AxCtp_ProxyFileInfo =
{
    (PCInterfaceProxyVtblList *) & _AxCtp_ProxyVtblList,
    (PCInterfaceStubVtblList *) & _AxCtp_StubVtblList,
    (const PCInterfaceName * ) & _AxCtp_InterfaceNamesList,
    (const IID ** ) & _AxCtp_BaseIIDList,
    & _AxCtp_IID_Lookup,
    1,
    2,
    0, /* table of [async_uuid] interfaces */
    0, /* Filler1 */
    0, /* Filler2 */
    0  /* Filler3 */
};
```

```
/* this file contains the actual definitions of */
/* the IIDs and CLSIDs */

/* link this file in with the server and any clients */


/* File created by MIDL compiler version 5.01.0164 */
/* at Thu Mar 09 10:57:11 2000
 */
/* Compiler settings for AxCtp.idl:
    Oicf (OptLev=i2), W1, Zp8, env=Win32, ms_ext, c_ext
    error checks: allocation ref bounds_check enum stub_data
*/
//@@MIDL_FILE_HEADING(  )
#ifdef __cplusplus
extern "C"{
#endif


#ifndef __IID_DEFINED__
#define __IID_DEFINED__

typedef struct _IID
{
    unsigned long x;
    unsigned short s1;
    unsigned short s2;
    unsigned char  c[8];
} IID;

#endif // __IID_DEFINED__

#ifndef CLSID_DEFINED
#define CLSID_DEFINED
typedef IID CLSID;
#endif // CLSID_DEFINED

const IID IID_ICtp = {0x38578BFE,0x0ABB,0x11D3,{0x93,0x30,0x00,0x80,0xC6,0xF7,0x96,0xA1}};


const IID LIBID_AXCTPLib = {0x38578BF1,0x0ABB,0x11D3,{0x93,0x30,0x00,0x80,0xC6,0xF7,0x96,0xA1}};


const CLSID CLSID_Ctp = {0x38578BF0,0x0ABB,0x11D3,{0x93,0x30,0x00,0x80,0xC6,0xF7,0x96,0xA1}};


#ifdef __cplusplus
}
#endif
```

```
//Microsoft Developer Studio generated resource script.
//
#include "resource.h"

#define APSTUDIO_READONLY_SYMBOLS
//////////////////////////////////////////////////////////////////////
///////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "winres.h"
//////////////////////////////////////////////////////////////////////
///////////
#undef APSTUDIO_READONLY_SYMBOLS

//////////////////////////////////////////////////////////////////////
///////////
// English (U.S.) resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
#pragma code_page(1252)
#endif //_WIN32

#ifdef APSTUDIO_INVOKED
//////////////////////////////////////////////////////////////////////
///////////
//
// TEXTINCLUDE
//

1 TEXTINCLUDE DISCARDABLE
BEGIN
    "resource.h\0"
END

2 TEXTINCLUDE DISCARDABLE
BEGIN
    "#include ""winres.h""\r\n"
END

3 TEXTINCLUDE DISCARDABLE
BEGIN
    "1 TYPELIB ""AxCtp.tlb""\r\n"
        "#include ""Res\\Version.rc2""\r\n"
    "\0"
END

#endif    // APSTUDIO_INVOKED

///////////////////////////////////////////////////////////////////
///////////
//
// REGISTRY
//

IDR_CTP                   REGISTRY DISCARDABLE    "Ctp.rgs"

///////////////////////////////////////////////////////////////////
///////////
//
// Dialog
//

IDD_CTLPANEL DIALOGEX 0, 0, 137, 169
STYLE WS_CHILD
EXSTYLE WS_EX_TRANSPARENT
FONT 12, "CAC Futura Casual", 0, 0, 0x1
BEGIN
    LTEXT           "View Card Panel",IDC_STATIC,3,1,122,8
    CONTROL         "Front",IDC_PAGE1,"Button",BS_AUTORADIOBUTTON,
10,8,73,10
    CONTROL         "Inside Left",IDC_PAGE2,"Button",BS_AUTORADIOB
UTTON,10,
                    15,73,10
    CONTROL         "Inside Right",IDC_PAGE3,"Button",BS_AUTORADIO
BUTTON,10,
                    23,73,10
    CONTROL         "Back",IDC_PAGE4,"Button",BS_AUTORADIOBUTTON,1
0,31,73,10
    LTEXT           "Font",IDC_STATIC,3,44,122,8,0,WS_EX_TRANSPARE
NT
    COMBOBOX        IDC_FONT,10,53,122,112,CBS_DROPDOWNLIST |
                    CBS_OWNERDRAWFIXED | CBS_SORT | CBS_HASSTRINGS
 |
                    WS_VSCROLL | WS_GROUP | WS_TABSTOP
    LTEXT           "Point Size",IDC_STATIC,3,68,122,8,0,WS_EX_TRA
NSPARENT
    COMBOBOX        IDC_PTSIZE,10,77,38,88,CBS_DROPDOWNLIST | WS_V
SCROLL |
                    WS_GROUP | WS_TABSTOP
    LTEXT           "Text Color",IDC_STATIC,3,92,122,8,0,WS_EX_TRA
NSPARENT
    COMBOBOX        IDC_COLOR,10,101,67,67,CBS_DROPDOWNLIST |
                    CBS_OWNERDRAWFIXED | WS_VSCROLL | WS_TABSTOP,
                    WS_EX_TRANSPARENT
    LTEXT           "Text Alignment",IDC_STATIC,3,117,122,8,0,
                    WS_EX_TRANSPARENT
    CONTROL         "Left",IDC_LEFT,"Button",BS_AUTORADIOBUTTON |
WS_GROUP |
                    WS_TABSTOP,10,124,48,10,WS_EX_TRANSPARENT
    CONTROL         "Center",IDC_CENTER,"Button",BS_AUTORADIOBUTTO
N,10,132,
                    48,10,WS_EX_TRANSPARENT
    CONTROL         "Right",IDC_RIGHT,"Button",BS_AUTORADIOBUTTON,
10,140,48,
                    10,WS_EX_TRANSPARENT
    PUSHBUTTON      "Print",IDC_PRINT,3,154,50,11
END

1538 DIALOG DISCARDABLE  32, 32, 287, 157
STYLE DS_MODALFRAME | DS_3DLOOK | DS_CONTEXTHELP | WS_POPUP | WS_V
ISIBLE |
    WS_CAPTION | WS_SYSMENU
CAPTION "Print"
FONT 8, "MS Sans Serif"
BEGIN
    GROUPBOX        "Printer",1075,8,4,272,84,WS_GROUP
    LTEXT           "&Name:",1093,16,20,36,8
    COMBOBOX        1139,52,18,152,152,CBS_DROPDOWNLIST | CBS_SORT
 |
                    WS_VSCROLL | WS_GROUP | WS_TABSTOP
    PUSHBUTTON      "&Properties",1025,212,17,60,14,WS_GROUP
    LTEXT           "Status:",1095,16,36,36,10,SS_NOPREFIX
    CONTROL         "",1099,"Static",SS_LEFTNOWORDWRAP | SS_NOPREF
IX |
                    WS_GROUP,52,36,224,10
    LTEXT           "Type:",1094,16,48,36,10,SS_NOPREFIX
    CONTROL         "",1098,"Static",SS_LEFTNOWORDWRAP | SS_NOPREF
IX |
                    WS_GROUP,52,48,224,10
    LTEXT           "Where:",1097,16,60,36,10,SS_NOPREFIX
    CONTROL         "",1101,"Static",SS_LEFTNOWORDWRAP | SS_NOPREF
IX |
                    WS_GROUP,52,60,224,10
    LTEXT           "Comment:",1096,16,72,36,10,SS_NOPREFIX
    CONTROL         "",1100,"Static",SS_LEFTNOWORDWRAP | SS_NOPREF
```

```
IX |
                    WS_GROUP,52,72,152,10
    CONTROL         "Print to fi&le",1040,"Button",BS_AUTOCHECKBOX
 |
                    WS_GROUP | WS_TABSTOP,212,70,64,12
    GROUPBOX        "Print Format",IDC_STATIC,8,93,136,39,WS_GROUP
    CONTROL         "Single-fold",IDC_SINGLEFOLD,"Button",BS_AUTOR
ADIOBUTTON,
                    15,104,80,10
    CONTROL         "Quarter-fold",IDC_QUARTERFOLD,"Button",
                    BS_AUTORADIOBUTTON,15,116,80,10
    LTEXT           "Number of &copies:",1092,162,105,68,8
    EDITTEXT        1154,234,103,32,12,ES_NUMBER | WS_GROUP
    DEFPUSHBUTTON   "OK",IDOK,180,137,48,14,WS_GROUP
    PUSHBUTTON      "Cancel",IDCANCEL,232,137,48,14
    GROUPBOX        "Copies",IDC_STATIC,152,93,128,39,WS_GROUP
    CONTROL         "Run Double-Sided Printing Test",IDC_DBLSIDE,"
Button",
                    BS_AUTOCHECKBOX | WS_TABSTOP,8,140,115,10
END

IDD_WAITDLG DIALOG DISCARDABLE  0, 0, 186, 44
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION
CAPTION "Printing"
FONT 8, "MS Sans Serif"
BEGIN
    CTEXT           "Your document is being printed.  Please wait.
..",
                    IDC_STATIC,6,14,173,8
END

IDD_DBLSIDEINTRO DIALOG DISCARDABLE  0, 0, 346, 105
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Double-Sided Printing Test"
FONT 8, "MS Sans Serif"
BEGIN
    LTEXT           "To help guide you through printing a single-f
old card on both sides of a page, some information needs to be gat
hered about the way paper feeds through your printer.",
                    IDC_STATIC,0,4,345,24
    LTEXT           "This print test will use one piece of paper.
 It will only need to be run once for a particular printer.",
                    IDC_STATIC,0,32,345,24
    LTEXT           "Click Next when you are ready to print the te
st page.",
                    IDC_STATIC,0,97,345,8
```

```
END

IDD_DBLSIDESTEP1 DIALOG DISCARDABLE  0, 0, 346, 105
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Double-Sided Printing Test"
FONT 8, "MS Sans Serif"
BEGIN
    LTEXT           "After the test page has printed, it must be p
rinted on once more to complete this test.",
                    IDC_STATIC,0,4,345,8
    LTEXT           "Please put the page back into the printer wit
h the printed side UP and the arrow pointing TOWARD the printer.",
                    IDC_STATIC,0,24,345,24
    LTEXT           "Click Next when you are ready to print.",IDC_
STATIC,0,
                    97,345,8
END

IDD_DBLSIDESTEP2 DIALOG DISCARDABLE  0, 0, 346, 105
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Double-Sided Printing Test"
FONT 8, "MS Sans Serif"
BEGIN
    LTEXT           "Please click on the option below that matches
 your printed page.",
                    IDC_STATIC,0,4,345,8
    CONTROL         "",IDC_FRAME1,"Static",SS_ETCHEDFRAME,0,20,56,
68
    CONTROL         "",IDC_FRAME2,"Static",SS_ETCHEDFRAME,66,20,56
,68
    CONTROL         "",IDC_FRAME3,"Static",SS_ETCHEDFRAME,132,20,1
02,68
    CONTROL         "",IDC_FRAME4,"Static",SS_ETCHEDFRAME,244,20,1
02,68
    CONTROL         210,IDC_STATIC,"Static",SS_BITMAP,8,26,40,49
    CONTROL         212,IDC_STATIC,"Static",SS_BITMAP,74,26,40,49
    CONTROL         208,IDC_STATIC,"Static",SS_BITMAP,140,26,40,49
    CONTROL         208,IDC_STATIC,"Static",SS_BITMAP,252,26,40,49
    CONTROL         209,IDC_STATIC,"Static",SS_BITMAP,186,26,40,49
    CONTROL         211,IDC_STATIC,"Static",SS_BITMAP,298,26,40,49
    LTEXT           "Front",IDC_STATIC,152,76,17,8
    LTEXT           "Back",IDC_STATIC,197,76,18,8
    LTEXT           "Front",IDC_STATIC,264,77,17,8
    LTEXT           "Back",IDC_STATIC,309,77,18,8
    LTEXT           "Click Next to continue.",IDC_STATIC,0,97,345,
8
```

```
END

IDD_DBLSIDEEND DIALOG DISCARDABLE  0, 0, 346, 105
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Double-Sided Printing Test"
FONT 8, "MS Sans Serif"
BEGIN
    LTEXT           "The double-sided printing test is complete.",
IDC_STATIC,
                    0,4,345,8
    LTEXT           "It is now time to print your card.",IDC_STATI
C,0,24,345,
                    8
    LTEXT           "Click Finish when you are ready.",IDC_STATIC,
0,97,345,8
END


/////////////////////////////////////////////////////////////////
///////////
//
// DESIGNINFO
//

#ifdef APSTUDIO_INVOKED
GUIDELINES DESIGNINFO DISCARDABLE
BEGIN
    IDD_WAITDLG, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 179
        TOPMARGIN, 7
        BOTTOMMARGIN, 54
    END
END
#endif    // APSTUDIO_INVOKED


/////////////////////////////////////////////////////////////////
///////////
//
// AGIMAGE
//

IDR_AGLOGO              AGIMAGE DISCARDABLE     "Res\\AGLogo.agi"
IDR_CPLOGO              AGIMAGE DISCARDABLE     "Res\\C&PLogo.agi"
```

```
//////////////////////////////////////////////////////////////////
///////////
//
// TTZ
//

IDR_CACFC                TTZ     DISCARDABLE     "Res\\Cacfc___.ttz
"

//////////////////////////////////////////////////////////////////
///////////
//
// Bitmap
//

IDB_1UP                  BITMAP  DISCARDABLE     "Res\\1up.bmp"
IDB_2UP                  BITMAP  DISCARDABLE     "Res\\2up.bmp"
IDB_3UP                  BITMAP  DISCARDABLE     "Res\\3up.bmp"
IDB_2DOWN                BITMAP  DISCARDABLE     "Res\\2down.bmp"
IDB_1UP2DOWN             BITMAP  DISCARDABLE     "Res\\1up2down.bmp
"

//////////////////////////////////////////////////////////////////
///////////
//
// String Table
//

STRINGTABLE DISCARDABLE
BEGIN
    IDS_PROJNAME            "AxCtp"
END

#endif    // English (U.S.) resources
//////////////////////////////////////////////////////////////////
///////////



#ifndef APSTUDIO_INVOKED
//////////////////////////////////////////////////////////////////
///////////
//
// Generated from the TEXTINCLUDE 3 resource.
//
```

```
1 TYPELIB "AxCtp.tlb"
#include "Res\Version.rc2"

//////////////////////////////////////////////////////////////////////////
///////////
#endif    // not APSTUDIO_INVOKED
```

```
<html>
<body>
<pre>
<h1>Build Log</h1>
<h3>
--------------------Configuration: AxCtp - Win32 Release--------------------
</h3>
<h3>Command Lines</h3>
Creating temporary file "c:\windows\TEMP\RSPC0F3.TMP" with contents
[
/nologo /Zp2 /MT /W3 /GX /O1 /I "..\ZLib" /I "..\Stonehnd" /D "WIN32" /D "NDEBUG" /D "_WINDOWS" /D "
_USRDLL" /D "_ATL_STATIC_REGISTRY" /Fp"Release/AxCtp.pch" /Yu"stdafx.h" /Fo"Release/" /Fd"Release/"
/FD /c
"C:\Work\CrtPrt\Axctp\Ctp.cpp"
]
Creating command line "cl.exe @c:\windows\TEMP\RSPC0F3.TMP"
Creating temporary file "c:\windows\TEMP\RSPC0F4.TMP" with contents
[
kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.lib shell32.lib ole32.lib oleau
t32.lib uuid.lib odbc32.lib odbccp32.lib comctl32.lib /nologo /subsystem:windows /dll /incremental:n
o /pdb:"Release/AxCtp.pdb" /machine:I386 /def:".\AxCtp.def" /out:"Release/AxCtp.dll" /implib:"Releas
e/AxCtp.lib"
.\Release\AGDC.obj
.\Release\AGDoc.obj
.\Release\AGLayer.obj
.\Release\AGMatrix.obj
.\Release\AGPage.obj
.\Release\AGSym.obj
.\Release\AGText.obj
.\Release\AxCtp.obj
.\Release\CtlPanel.obj
.\Release\Ctp.obj
.\Release\dblside.obj
.\Release\Font.obj
.\Release\StdAfx.obj
.\Release\WaitDlg.obj
.\Release\AxCtp.res
\Work\CrtPrt\ZLib\Release\ZLib.lib
\Work\CrtPrt\Stonehnd\Release\Stonehnd.lib
]
Creating command line "link.exe @c:\windows\TEMP\RSPC0F4.TMP"
Creating temporary file "c:\windows\TEMP\RSPC0F5.BAT" with contents
[
@echo off
regsvr32 /s /c ".\Release\AxCtp.dll"
echo regsvr32 exec. time > ".\Release\regsvr32.trg"
]
Creating command line "c:\windows\TEMP\RSPC0F5.BAT"
Compiling...
Ctp.cpp
Linking...
   Creating library Release/AxCtp.lib and object Release/AxCtp.exp
<h3>Output Window</h3>
Registering ActiveX Control...



<h3>Results</h3>
AxCtp.dll - 0 error(s), 0 warning(s)
</pre>
</body>
</html>
```

```
#include <olectl.h>
// AxCtp.idl : IDL source for AxCtp.dll
//

// This file will be processed by the MIDL tool to
// produce the type library (AxCtp.tlb) and marshalling code.

import "oaidl.idl";
import "ocidl.idl";



    [
        object,
        uuid(38578BFE-0ABB-11D3-9330-0080C6F796A1),
        dual,
        helpstring("ICtp Interface"),
        pointer_default(unique)
    ]
    interface ICtp : IDispatch
    {
        [propput, id(0)]
        HRESULT Fonts([in]BSTR strFonts);
        [propget, id(0)]
        HRESULT Fonts([out,retval]BSTR* pstrFonts);

        [propput, id(1)]
        HRESULT Src([in]BSTR strSrc);
        [propget, id(1)]
        HRESULT Src([out,retval]BSTR* pstrSrc);
    };
[
    uuid(38578BF1-0ABB-11D3-9330-0080C6F796A1),
    version(1.0),
    helpstring("AxCtp 1.0 Type Library")
]
library AXCTPLib
{
    importlib("stdole32.tlb");
    importlib("stdole2.tlb");

    [
        uuid(38578BF0-0ABB-11D3-9330-0080C6F796A1),
        helpstring("Ctp Class")
    ]
    coclass Ctp
    {
        [default] interface ICtp;
    };
};
```

```
/* this ALWAYS GENERATED file contains the definitions for the interfaces */


/* File created by MIDL compiler version 5.01.0164 */
/* at Thu Mar 09 10:57:11 2000
 */
/* Compiler settings for AxCtp.idl:
    Oicf (OptLev=i2), W1, Zp8, env=Win32, ms_ext, c_ext
    error checks: allocation ref bounds_check enum stub_data
*/
//@@MIDL_FILE_HEADING(  )


/* verify that the <rpcndr.h> version is high enough to compile this file*/
#ifndef __REQUIRED_RPCNDR_H_VERSION__
#define __REQUIRED_RPCNDR_H_VERSION__ 440
#endif

#include "rpc.h"
#include "rpcndr.h"

#ifndef __RPCNDR_H_VERSION__
#error this stub requires an updated version of <rpcndr.h>
#endif // __RPCNDR_H_VERSION__

#ifndef COM_NO_WINDOWS_H
#include "windows.h"
#include "ole2.h"
#endif /*COM_NO_WINDOWS_H*/

#ifndef __AxCtp_h__
#define __AxCtp_h__

#ifdef __cplusplus
extern "C"{
#endif

/* Forward Declarations */

#ifndef __ICtp_FWD_DEFINED__
#define __ICtp_FWD_DEFINED__
typedef interface ICtp ICtp;
#endif  /* __ICtp_FWD_DEFINED__ */


#ifndef __Ctp_FWD_DEFINED__
#define __Ctp_FWD_DEFINED__

#ifdef __cplusplus
typedef class Ctp Ctp;
#else
typedef struct Ctp Ctp;
#endif /* __cplusplus */

#endif  /* __Ctp_FWD_DEFINED__ */


/* header files for imported files */
#include "oaidl.h"
#include "ocidl.h"

void __RPC_FAR * __RPC_USER MIDL_user_allocate(size_t);
void __RPC_USER MIDL_user_free( void __RPC_FAR * );

#ifndef __ICtp_INTERFACE_DEFINED__
#define __ICtp_INTERFACE_DEFINED__

/* interface ICtp */
/* [unique][helpstring][dual][uuid][object] */


EXTERN_C const IID IID_ICtp;
```

```
#if defined(__cplusplus) && !defined(CINTERFACE)

    MIDL_INTERFACE("38578BFE-0ABB-11D3-9330-0080C6F796A1")
    ICtp : public IDispatch
    {
    public:
        virtual /* [id][propput] */ HRESULT STDMETHODCALLTYPE put_Fonts(
            /* [in] */ BSTR strFonts) = 0;

        virtual /* [id][propget] */ HRESULT STDMETHODCALLTYPE get_Fonts(
            /* [retval][out] */ BSTR __RPC_FAR *pstrFonts) = 0;

        virtual /* [id][propput] */ HRESULT STDMETHODCALLTYPE put_Src(
            /* [in] */ BSTR strSrc) = 0;

        virtual /* [id][propget] */ HRESULT STDMETHODCALLTYPE get_Src(
            /* [retval][out] */ BSTR __RPC_FAR *pstrSrc) = 0;

    };

#else   /* C style interface */

    typedef struct ICtpVtbl
    {
        BEGIN_INTERFACE

        HRESULT ( STDMETHODCALLTYPE __RPC_FAR *QueryInterface )(
            ICtp __RPC_FAR * This,
            /* [in] */ REFIID riid,
            /* [iid_is][out] */ void __RPC_FAR *__RPC_FAR *ppvObject);

        ULONG ( STDMETHODCALLTYPE __RPC_FAR *AddRef )(
            ICtp __RPC_FAR * This);

        ULONG ( STDMETHODCALLTYPE __RPC_FAR *Release )(
            ICtp __RPC_FAR * This);

        HRESULT ( STDMETHODCALLTYPE __RPC_FAR *GetTypeInfoCount )(
            ICtp __RPC_FAR * This,
            /* [out] */ UINT __RPC_FAR *pctinfo);

        HRESULT ( STDMETHODCALLTYPE __RPC_FAR *GetTypeInfo )(
            ICtp __RPC_FAR * This,
            /* [in] */ UINT iTInfo,
            /* [in] */ LCID lcid,
            /* [out] */ ITypeInfo __RPC_FAR *__RPC_FAR *ppTInfo);

        HRESULT ( STDMETHODCALLTYPE __RPC_FAR *GetIDsOfNames )(
            ICtp __RPC_FAR * This,
            /* [in] */ REFIID riid,
            /* [size_is][in] */ LPOLESTR __RPC_FAR *rgszNames,
            /* [in] */ UINT cNames,
            /* [in] */ LCID lcid,
            /* [size_is][out] */ DISPID __RPC_FAR *rgDispId);

        /* [local] */ HRESULT ( STDMETHODCALLTYPE __RPC_FAR *Invoke )(
            ICtp __RPC_FAR * This,
            /* [in] */ DISPID dispIdMember,
            /* [in] */ REFIID riid,
            /* [in] */ LCID lcid,
            /* [in] */ WORD wFlags,
            /* [out][in] */ DISPPARAMS __RPC_FAR *pDispParams,
            /* [out] */ VARIANT __RPC_FAR *pVarResult,
            /* [out] */ EXCEPINFO __RPC_FAR *pExcepInfo,
            /* [out] */ UINT __RPC_FAR *puArgErr);

        /* [id][propput] */ HRESULT ( STDMETHODCALLTYPE __RPC_FAR *put_Fonts )(
            ICtp __RPC_FAR * This,
            /* [in] */ BSTR strFonts);

        /* [id][propget] */ HRESULT ( STDMETHODCALLTYPE __RPC_FAR *get_Fonts )(
            ICtp __RPC_FAR * This,
            /* [retval][out] */ BSTR __RPC_FAR *pstrFonts);
        /* [id][propput] */ HRESULT ( STDMETHODCALLTYPE __RPC_FAR *put_Src )(
            ICtp __RPC_FAR * This,
            /* [in] */ BSTR strSrc);

        /* [id][propget] */ HRESULT ( STDMETHODCALLTYPE __RPC_FAR *get_Src )(
            ICtp __RPC_FAR * This,
            /* [retval][out] */ BSTR __RPC_FAR *pstrSrc);

        END_INTERFACE
    } ICtpVtbl;

    interface ICtp
    {
        CONST_VTBL struct ICtpVtbl __RPC_FAR *lpVtbl;
    };


#ifdef COBJMACROS


#define ICtp_QueryInterface(This,riid,ppvObject)    \
    (This)->lpVtbl -> QueryInterface(This,riid,ppvObject)

#define ICtp_AddRef(This)   \
    (This)->lpVtbl -> AddRef(This)

#define ICtp_Release(This)  \
    (This)->lpVtbl -> Release(This)


#define ICtp_GetTypeInfoCount(This,pctinfo) \
    (This)->lpVtbl -> GetTypeInfoCount(This,pctinfo)

#define ICtp_GetTypeInfo(This,iTInfo,lcid,ppTInfo)  \
    (This)->lpVtbl -> GetTypeInfo(This,iTInfo,lcid,ppTInfo)

#define ICtp_GetIDsOfNames(This,riid,rgszNames,cNames,lcid,rgDispId)    \
    (This)->lpVtbl -> GetIDsOfNames(This,riid,rgszNames,cNames,lcid,rgDispId)

#define ICtp_Invoke(This,dispIdMember,riid,lcid,wFlags,pDispParams,pVarResult,pExcepInfo,puArgErr)
\
    (This)->lpVtbl -> Invoke(This,dispIdMember,riid,lcid,wFlags,pDispParams,pVarResult,pExcepInfo,pu
ArgErr)


#define ICtp_put_Fonts(This,strFonts)   \
    (This)->lpVtbl -> put_Fonts(This,strFonts)

#define ICtp_get_Fonts(This,pstrFonts)  \
    (This)->lpVtbl -> get_Fonts(This,pstrFonts)

#define ICtp_put_Src(This,strSrc)   \
    (This)->lpVtbl -> put_Src(This,strSrc)

#define ICtp_get_Src(This,pstrSrc)  \
    (This)->lpVtbl -> get_Src(This,pstrSrc)

#endif /* COBJMACROS */


#endif  /* C style interface */



/* [id][propput] */ HRESULT STDMETHODCALLTYPE ICtp_put_Fonts_Proxy(
    ICtp __RPC_FAR * This,
    /* [in] */ BSTR strFonts);


void __RPC_STUB ICtp_put_Fonts_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *_pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);


/* [id][propget] */ HRESULT STDMETHODCALLTYPE ICtp_get_Fonts_Proxy(
    ICtp __RPC_FAR * This,
    /* [retval][out] */ BSTR __RPC_FAR *pstrFonts);


void __RPC_STUB ICtp_get_Fonts_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *_pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);


/* [id][propput] */ HRESULT STDMETHODCALLTYPE ICtp_put_Src_Proxy(
    ICtp __RPC_FAR * This,
    /* [in] */ BSTR strSrc);


void __RPC_STUB ICtp_put_Src_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *_pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);


/* [id][propget] */ HRESULT STDMETHODCALLTYPE ICtp_get_Src_Proxy(
    ICtp __RPC_FAR * This,
    /* [retval][out] */ BSTR __RPC_FAR *pstrSrc);


void __RPC_STUB ICtp_get_Src_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *_pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);



#endif  /* __ICtp_INTERFACE_DEFINED__ */


#ifndef __AXCTPLib_LIBRARY_DEFINED__
#define __AXCTPLib_LIBRARY_DEFINED__

/* library AXCTPLib */
/* [helpstring][version][uuid] */


EXTERN_C const IID LIBID_AXCTPLib;

EXTERN_C const CLSID CLSID_Ctp;

#ifdef __cplusplus

class DECLSPEC_UUID("38578BF0-0ABB-11D3-9330-0080C6F796A1")
Ctp;
#endif
#endif /* __AXCTPLib_LIBRARY_DEFINED__ */

/* Additional Prototypes for ALL interfaces */

unsigned long             __RPC_USER  BSTR_UserSize(     unsigned long __RPC_FAR *, unsigned long
        , BSTR __RPC_FAR * );
unsigned char __RPC_FAR * __RPC_USER  BSTR_UserMarshal(  unsigned long __RPC_FAR *, unsigned char __
RPC_FAR *, BSTR __RPC_FAR * );
unsigned char __RPC_FAR * __RPC_USER  BSTR_UserUnmarshal(unsigned long __RPC_FAR *, unsigned char __
RPC_FAR *, BSTR __RPC_FAR * );
void                      __RPC_USER  BSTR_UserFree(     unsigned long __RPC_FAR *, BSTR __RPC_FAR *
 );

/* end of Additional Prototypes */

#ifdef __cplusplus
}
#endif

#endif
```

09629370.073400

```
Microsoft Developer Studio Workspace File, Format Version 6.00
# WARNING: DO NOT EDIT OR DELETE THIS WORKSPACE FILE!

###################################################################
#############

Project: "AxCtp"=.\AxCtp.dsp - Package Owner=<4>

Package=<5>
{{{
    begin source code control
    "$/AxCtp", NDAAAAAA
    .
    end source code control
}}}

Package=<4>
{{{
    Begin Project Dependency
    Project_Dep_Name ZLib
    End Project Dependency
    Begin Project Dependency
    Project_Dep_Name Stonehnd
    End Project Dependency
}}}

###################################################################
#############

Project: "Stonehnd"=..\Stonehnd\Stonehnd.dsp - Package Owner=<4>

Package=<5>
{{{
    begin source code control
    "$/Stonehnd", CGAAAAAA
    ..\stonehnd
    end source code control
}}}

Package=<4>
{{{
}}}

###################################################################
#############
```

```
Project: "ZLib"=..\ZLib\ZLib.dsp - Package Owner=<4>

Package=<5>
{{{
    begin source code control
    "$/ZLib", LBAAAAAA
    ..\zlib
    end source code control
}}}

Package=<4>
{{{
}}}

###################################################################
#############

Global:

Package=<5>
{{{
    begin source code control
    "$/AxCtp", NDAAAAAA
    .
    end source code control
}}}

Package=<3>
{{{
}}}

###################################################################
#############
```

```
# Microsoft Developer Studio Project File - Name="AxCtp" - Package
 Owner=<4>
# Microsoft Developer Studio Generated Build File, Format Version
6.00
# ** DO NOT EDIT **

# TARGTYPE "Win32 (x86) Dynamic-Link Library" 0x0102

CFG=AxCtp - Win32 Debug
!MESSAGE This is not a valid makefile. To build this project using
 NMAKE,
!MESSAGE use the Export Makefile command and run
!MESSAGE
!MESSAGE NMAKE /f "AxCtp.mak".
!MESSAGE
!MESSAGE You can specify a configuration when running NMAKE
!MESSAGE by defining the macro CFG on the command line. For exampl
e:
!MESSAGE
!MESSAGE NMAKE /f "AxCtp.mak" CFG="AxCtp - Win32 Debug"
!MESSAGE
!MESSAGE Possible choices for configuration are:
!MESSAGE
!MESSAGE "AxCtp - Win32 Debug" (based on "Win32 (x86) Dynamic-Link
 Library")
!MESSAGE "AxCtp - Win32 Release" (based on "Win32 (x86) Dynamic-Li
nk Library")
!MESSAGE

# Begin Project
# PROP AllowPerConfigDependencies 0
# PROP Scc_ProjName ""$/AxCtp", NDAAAAAA"
# PROP Scc_LocalPath "."
CPP=cl.exe
MTL=midl.exe
RSC=rc.exe

!IF  "$(CFG)" == "AxCtp - Win32 Debug"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "Debug"
# PROP BASE Intermediate_Dir "Debug"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Debug"
# PROP Intermediate_Dir "Debug"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MTd /W3 /Gm /Zi /Od /D "WIN32" /D "_DEBUG"
 /D "_WINDOWS" /D "_USRDLL" /Yu"stdafx.h" /FD /c
# ADD CPP /nologo /Zp2 /MTd /W3 /Gm /GX /ZI /Od /I "..\ZLib" /I ".
.\Stonehnd" /D "WIN32" /D "_DEBUG" /D "_WINDOWS" /D "_USRDLL" /D "
_ATL_STATIC_REGISTRY" /Yu"stdafx.h" /FD /c
# ADD BASE MTL /nologo /D "_DEBUG" /mktyplib203 /o "NUL" /win32
# ADD MTL /nologo /D "_DEBUG" /mktyplib203 /o "NUL" /win32
# ADD BASE RSC /l 0x409 /d "_DEBUG"
# ADD RSC /l 0x409 /d "_DEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib c
omdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.l
ib odbc32.lib odbccp32.lib /nologo /subsystem:windows /dll /debug
/machine:I386 /pdbtype:sept
# ADD LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg
32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib od
bc32.lib odbccp32.lib comctl32.lib /nologo /subsystem:windows /dll
 /debug /machine:I386 /pdbtype:sept
# Begin Custom Build - Registering ActiveX Control...
OutDir=.\Debug
TargetPath=.\Debug\AxCtp.dll
InputPath=.\Debug\AxCtp.dll
SOURCE="$(InputPath)"

"$(OutDir)\regsvr32.trg" : $(SOURCE) "$(INTDIR)" "$(OUTDIR)"
	regsvr32 /s /c "$(TargetPath)"
	echo regsvr32 exec. time > "$(OutDir)\regsvr32.trg"

# End Custom Build

!ELSEIF  "$(CFG)" == "AxCtp - Win32 Release"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "AxCtp___"
# PROP BASE Intermediate_Dir "AxCtp___"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Release"
# PROP Intermediate_Dir "Release"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MT /W3 /O1 /D "WIN32" /D "NDEBUG" /D "_WIN
DOWS" /D "_USRDLL" /D "_ATL_STATIC_REGISTRY" /D "_ATL_MIN_CRT" /Yu
"stdafx.h" /FD /c
# ADD CPP /nologo /Zp2 /MT /W3 /GX /O1 /I "..\ZLib" /I "..\Stonehn
d" /D "WIN32" /D "NDEBUG" /D "_WINDOWS" /D "_USRDLL" /D "_ATL_STAT
IC_REGISTRY" /Yu"stdafx.h" /FD /c
# ADD BASE MTL /nologo /D "NDEBUG" /mktyplib203 /o "NUL" /win32
# ADD MTL /nologo /D "NDEBUG" /mktyplib203 /o "NUL" /win32
# ADD BASE RSC /l 0x409 /d "NDEBUG"
# ADD RSC /l 0x409 /d "NDEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib c
omdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.l
ib odbc32.lib odbccp32.lib /nologo /subsystem:windows /dll /machin
e:I386
# ADD LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg
32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib od
bc32.lib odbccp32.lib comctl32.lib /nologo /subsystem:windows /dll
 /machine:I386
# Begin Custom Build - Registering ActiveX Control...
OutDir=.\Release
TargetPath=.\Release\AxCtp.dll
InputPath=.\Release\AxCtp.dll
SOURCE="$(InputPath)"

"$(OutDir)\regsvr32.trg" : $(SOURCE) "$(INTDIR)" "$(OUTDIR)"
	regsvr32 /s /c "$(TargetPath)"
	echo regsvr32 exec. time > "$(OutDir)\regsvr32.trg"

# End Custom Build

!ENDIF

# Begin Target

# Name "AxCtp - Win32 Debug"
# Name "AxCtp - Win32 Release"
# Begin Group "Source Files"
# PROP Default_Filter "cpp;c;cxx;rc;def;r;odl;idl;hpj;bat"
# Begin Source File

SOURCE=.\AGDC.cpp
# End Source File
# Begin Source File

SOURCE=.\AGDoc.cpp
# End Source File
# Begin Source File

SOURCE=.\AGLayer.cpp
# End Source File
# Begin Source File

SOURCE=.\AGMatrix.cpp
# End Source File
# Begin Source File

SOURCE=.\AGPage.cpp
# End Source File
# Begin Source File

SOURCE=.\AGSym.cpp
# End Source File
# Begin Source File

SOURCE=.\AGText.cpp
# End Source File
# Begin Source File

SOURCE=.\AxCtp.cpp
# End Source File
# Begin Source File

SOURCE=.\AxCtp.def
# End Source File
# Begin Source File

SOURCE=.\AxCtp.idl

!IF  "$(CFG)" == "AxCtp - Win32 Debug"

# PROP Ignore_Default_Tool 1
# Begin Custom Build - Performing MIDL step
InputPath=.\AxCtp.idl
```

```
BuildCmds= \
	midl /Oicf /h "AxCtp.h" /iid "AxCtp_i.c" "AxCtp.idl"

".\AxCtp.tlb" : $(SOURCE) "$(INTDIR)" "$(OUTDIR)"
   $(BuildCmds)

".\AxCtp.h" : $(SOURCE) "$(INTDIR)" "$(OUTDIR)"
   $(BuildCmds)

".\AxCtp_i.c" : $(SOURCE) "$(INTDIR)" "$(OUTDIR)"
   $(BuildCmds)
# End Custom Build

!ELSEIF  "$(CFG)" == "AxCtp - Win32 Release"

# PROP Ignore_Default_Tool 1
# Begin Custom Build - Performing MIDL step
InputPath=.\AxCtp.idl

BuildCmds= \
	midl /Oicf /h "AxCtp.h" /iid "AxCtp_i.c" "AxCtp.idl"

".\AxCtp.tlb" : $(SOURCE) "$(INTDIR)" "$(OUTDIR)"
   $(BuildCmds)

".\AxCtp.h" : $(SOURCE) "$(INTDIR)" "$(OUTDIR)"
   $(BuildCmds)

".\AxCtp_i.c" : $(SOURCE) "$(INTDIR)" "$(OUTDIR)"
   $(BuildCmds)
# End Custom Build

!ENDIF

# End Source File
# Begin Source File

SOURCE=.\AxCtp.rc
# End Source File
# Begin Source File

SOURCE=.\CtlPanel.cpp
# End Source File
# Begin Source File
```

```
SOURCE=.\Ctp.cpp
# End Source File
# Begin Source File

SOURCE=.\dblside.cpp
# End Source File
# Begin Source File

SOURCE=.\Font.cpp
# End Source File
# Begin Source File

SOURCE=.\StdAfx.cpp
# ADD CPP /Yc"stdafx.h"
# End Source File
# Begin Source File

SOURCE=.\WaitDlg.cpp
# End Source File
# End Group
# Begin Group "Header Files"

# PROP Default_Filter "h;hpp;hxx;hm;inl"
# Begin Source File

SOURCE=.\AGDC.h
# End Source File
# Begin Source File

SOURCE=.\AGDib.h
# End Source File
# Begin Source File

SOURCE=.\AGDoc.h
# End Source File
# Begin Source File

SOURCE=.\AGLayer.h
# End Source File
# Begin Source File

SOURCE=.\AGMatrix.h
# End Source File
# Begin Source File

SOURCE=.\AGPage.h
```

```
# End Source File
# Begin Source File

SOURCE=.\AGSym.h
# End Source File
# Begin Source File

SOURCE=.\AGText.h
# End Source File
# Begin Source File

SOURCE=.\Bsc2.h
# End Source File
# Begin Source File

SOURCE=.\CtlPanel.h
# End Source File
# Begin Source File

SOURCE=.\Ctp.h
# End Source File
# Begin Source File

SOURCE=.\dblside.h
# End Source File
# Begin Source File

SOURCE=.\Font.h
# End Source File
# Begin Source File

SOURCE=.\propsht.h
# End Source File
# Begin Source File

SOURCE=.\Resource.h
# End Source File
# Begin Source File

SOURCE=.\StdAfx.h
# End Source File
# Begin Source File

SOURCE=.\version.h
# End Source File
# Begin Source File
```

```
SOURCE=.\WaitDlg.h
# End Source File
# End Group
# Begin Group "Resource Files"

# PROP Default_Filter "ico;cur;bmp;dlg;rc2;rct;bin;cnt;rtf;gif;jpg
;jpeg;jpe"
# Begin Source File

SOURCE=.\Res\1up.bmp
# End Source File
# Begin Source File

SOURCE=.\Res\1up2down.bmp
# End Source File
# Begin Source File

SOURCE=.\Res\2down.bmp
# End Source File
# Begin Source File

SOURCE=.\Res\2up.bmp
# End Source File
# Begin Source File

SOURCE=.\Res\3up.bmp
# End Source File
# Begin Source File

SOURCE=.\Res\AGLogo.agi
# End Source File
# Begin Source File

SOURCE=".\Res\C&PLogo.agi"
# End Source File
# Begin Source File

SOURCE=.\Res\Cacfc___.ttz
# End Source File
# Begin Source File

SOURCE=.\Ctp.rgs
# End Source File
# Begin Source File
```

```
SOURCE=.\Res\Version.rc2
# End Source File
# End Group
# End Target
# End Project
```

```
; AxCtp.def : Declares the module parameters.

LIBRARY      "AxCtp.DLL"

EXPORTS
    DllCanUnloadNow     @1 PRIVATE
    DllGetClassObject   @2 PRIVATE
    DllRegisterServer   @3 PRIVATE
    DllUnregisterServer @4 PRIVATE
```

```
//===========================================================================//
//===========================================================================//
#include "stdafx.h"
#include "resource.h"
#include "initguid.h"
#include "AxCtp.h"

#include "AxCtp_i.c"
#include "Ctp.h"

#include "scappint.h"


CComModule _Module;

BEGIN_OBJECT_MAP(ObjectMap)
    OBJECT_ENTRY(CLSID_Ctp, CCtp)
END_OBJECT_MAP()

/////////////////////////////////////////////////////////////////////////////
// DLL Entry Point

extern "C"
BOOL WINAPI DllMain(HINSTANCE hInstance, DWORD dwReason, LPVOID /*lpReserved*/)
{
    if (dwReason == DLL_PROCESS_ATTACH)
    {
        _Module.Init(ObjectMap, hInstance);
        DisableThreadLibraryCalls(hInstance);
        SCENG_Init();
    }
    else if (dwReason == DLL_PROCESS_DETACH)
    {
        _Module.Term();
        CAGDC::Free();
        SCENG_Fini();
    }
    return TRUE;    // ok
}

/////////////////////////////////////////////////////////////////////////////
// Used to determine whether the DLL can be unloaded by OLE

STDAPI DllCanUnloadNow(void)
{
    return (_Module.GetLockCount()==0) ? S_OK : S_FALSE;
}

/////////////////////////////////////////////////////////////////////////////
// Returns a class factory to create an object of the requested type

STDAPI DllGetClassObject(REFCLSID rclsid, REFIID riid, LPVOID* ppv)
{
    return _Module.GetClassObject(rclsid, riid, ppv);
}

/////////////////////////////////////////////////////////////////////////////
// DllRegisterServer - Adds entries to the system registry

STDAPI DllRegisterServer(void)
{
    // registers object, typelib and all interfaces in typelib
    return _Module.RegisterServer(TRUE);
}

/////////////////////////////////////////////////////////////////////////////
// DllUnregisterServer - Removes entries from the system registry

STDAPI DllUnregisterServer(void)
{
    _Module.UnregisterServer();
    return S_OK;
}
```

```
# Microsoft Developer Studio Project File - Name="AxCtp" - Package
 Owner=<4>
# Microsoft Developer Studio Generated Build File, Format Version
5.00
# ** DO NOT EDIT **

# TARGTYPE "Win32 (x86) Dynamic-Link Library" 0x0102

CFG=AxCtp - Win32 Debug
!MESSAGE This is not a valid makefile. To build this project using
 NMAKE,
!MESSAGE use the Export Makefile command and run
!MESSAGE
!MESSAGE NMAKE /f "AxCtp.mak".
!MESSAGE
!MESSAGE You can specify a configuration when running NMAKE
!MESSAGE by defining the macro CFG on the command line. For exampl
e:
!MESSAGE
!MESSAGE NMAKE /f "AxCtp.mak" CFG="AxCtp - Win32 Debug"
!MESSAGE
!MESSAGE Possible choices for configuration are:
!MESSAGE
!MESSAGE "AxCtp - Win32 Debug" (based on "Win32 (x86) Dynamic-Link
 Library")
!MESSAGE "AxCtp - Win32 Release" (based on "Win32 (x86) Dynamic-Li
nk Library")
!MESSAGE

# Begin Project
# PROP Scc_ProjName ""$/AxCtp", NDAAAAAA"
# PROP Scc_LocalPath "."
CPP=cl.exe
MTL=midl.exe
RSC=rc.exe

!IF  "$(CFG)" == "AxCtp - Win32 Debug"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "Debug"
# PROP BASE Intermediate_Dir "Debug"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Debug"
# PROP Intermediate_Dir "Debug"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MTd /W3 /Gm /Zi /Od /D "WIN32" /D "_DEBUG"
 /D "_WINDOWS" /D "_USRDLL" /Yu"stdafx.h" /FD /c
# ADD CPP /nologo /Zp2 /MTd /W3 /Gm /GX /Zi /Od /I "..\ZLib" /I ".
.\Stonehnd" /D "WIN32" /D "_DEBUG" /D "_WINDOWS" /D "_USRDLL" /D "
_ATL_STATIC_REGISTRY" /Yu"stdafx.h" /FD /c
# ADD BASE MTL /nologo /D "_DEBUG" /mktyplib203 /o NUL /win32
# ADD MTL /nologo /D "_DEBUG" /mktyplib203 /o NUL /win32
# ADD BASE RSC /l 0x409 /d "_DEBUG"
# ADD RSC /l 0x409 /d "_DEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib c
omdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.l
ib odbc32.lib odbccp32.lib /nologo /subsystem:windows /dll /debug
/machine:I386 /pdbtype:sept
# ADD LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg
32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib od
bc32.lib odbccp32.lib comctl32.lib /nologo /subsystem:windows /dll
 /debug /machine:I386 /pdbtype:sept
# Begin Custom Build - Registering ActiveX Control...
OutDir=.\Debug
TargetPath=.\Debug\AxCtp.dll
InputPath=.\Debug\AxCtp.dll
SOURCE=$(InputPath)

"$(OutDir)\regsvr32.trg" : $(SOURCE) "$(INTDIR)" "$(OUTDIR)"
	regsvr32 /s /c "$(TargetPath)"
	echo regsvr32 exec. time > "$(OutDir)\regsvr32.trg"

# End Custom Build

!ELSEIF  "$(CFG)" == "AxCtp - Win32 Release"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "AxCtp___"
# PROP BASE Intermediate_Dir "AxCtp___"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Release"
# PROP Intermediate_Dir "Release"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MT /W3 /O1 /D "WIN32" /D "NDEBUG" /D "_WIN
DOWS" /D "_USRDLL" /D "_ATL_STATIC_REGISTRY" /D "_ATL_MIN_CRT" /Yu
"stdafx.h" /FD /c
# ADD CPP /nologo /Zp2 /MT /W3 /GX /O1 /I "..\ZLib" /I "..\Stonehn
d" /D "WIN32" /D "NDEBUG" /D "_WINDOWS" /D "_USRDLL" /D "_ATL_STAT
IC_REGISTRY" /Yu"stdafx.h" /FD /c
# ADD BASE MTL /nologo /D "NDEBUG" /mktyplib203 /o NUL /win32
# ADD MTL /nologo /D "NDEBUG" /mktyplib203 /o NUL /win32
# ADD BASE RSC /l 0x409 /d "NDEBUG"
# ADD RSC /l 0x409 /d "NDEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib c
omdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.l
ib odbc32.lib odbccp32.lib /nologo /subsystem:windows /dll /machin
e:I386
# ADD LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg
32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib od
bc32.lib odbccp32.lib comctl32.lib /nologo /subsystem:windows /dll
 /machine:I386
# Begin Custom Build - Registering ActiveX Control...
OutDir=.\Release
TargetPath=.\Release\AxCtp.dll
InputPath=.\Release\AxCtp.dll
SOURCE=$(InputPath)

"$(OutDir)\regsvr32.trg" : $(SOURCE) "$(INTDIR)" "$(OUTDIR)"
	regsvr32 /s /c "$(TargetPath)"
	echo regsvr32 exec. time > "$(OutDir)\regsvr32.trg"

# End Custom Build

!ENDIF

# Begin Target

# Name "AxCtp - Win32 Debug"
# Name "AxCtp - Win32 Release"
# Begin Group "Source Files"

# PROP Default_Filter "cpp;c;cxx;rc;def;r;odl;idl;hpj;bat"
# Begin Source File

SOURCE=.\AGDC.cpp
# End Source File
# Begin Source File

SOURCE=.\AGDoc.cpp
# End Source File
# Begin Source File

SOURCE=.\AGLayer.cpp
# End Source File
# Begin Source File

SOURCE=.\AGMatrix.cpp
# End Source File
# Begin Source File

SOURCE=.\AGPage.cpp
# End Source File
# Begin Source File

SOURCE=.\AGSym.cpp
# End Source File
# Begin Source File

SOURCE=.\AGText.cpp
# End Source File
# Begin Source File

SOURCE=.\AxCtp.cpp
# End Source File
# Begin Source File

SOURCE=.\AxCtp.def
# End Source File
# Begin Source File

SOURCE=.\AxCtp.idl

!IF  "$(CFG)" == "AxCtp - Win32 Debug"

# Begin Custom Build - Performing MIDL step
InputPath=.\AxCtp.idl

BuildCmds= \
        midl /Oicf /h "AxCtp.h" /iid "AxCtp_i.c" "AxCtp.idl"

".\AxCtp.tlb" : $(SOURCE) "$(INTDIR)" "$(OUTDIR)"
   $(BuildCmds)

".\AxCtp.h" : $(SOURCE) "$(INTDIR)" "$(OUTDIR)"
   $(BuildCmds)

".\AxCtp_i.c" : $(SOURCE) "$(INTDIR)" "$(OUTDIR)"
   $(BuildCmds)
# End Custom Build

!ELSEIF  "$(CFG)" == "AxCtp - Win32 Release"

# Begin Custom Build - Performing MIDL step
InputPath=.\AxCtp.idl

BuildCmds= \
        midl /Oicf /h "AxCtp.h" /iid "AxCtp_i.c" "AxCtp.idl"

".\AxCtp.tlb" : $(SOURCE) "$(INTDIR)" "$(OUTDIR)"
   $(BuildCmds)

".\AxCtp.h" : $(SOURCE) "$(INTDIR)" "$(OUTDIR)"
   $(BuildCmds)

".\AxCtp_i.c" : $(SOURCE) "$(INTDIR)" "$(OUTDIR)"
   $(BuildCmds)
# End Custom Build

!ENDIF

# End Source File
# Begin Source File

SOURCE=.\AxCtp.rc
# End Source File
# Begin Source File

SOURCE=.\CtlPanel.cpp
# End Source File
# Begin Source File

SOURCE=.\Ctp.cpp
# End Source File
# Begin Source File
```

```
SOURCE=.\dblside.cpp
# End Source File
# Begin Source File

SOURCE=.\Font.cpp
# End Source File
# Begin Source File

SOURCE=.\StdAfx.cpp
# ADD CPP /Yc"stdafx.h"
# End Source File
# Begin Source File

SOURCE=.\WaitDlg.cpp
# End Source File
# End Group
# Begin Group "Header Files"

# PROP Default_Filter "h;hpp;hxx;hm;inl"
# Begin Source File

SOURCE=.\AGDC.h
# End Source File
# Begin Source File

SOURCE=.\AGDib.h
# End Source File
# Begin Source File

SOURCE=.\AGDoc.h
# End Source File
# Begin Source File

SOURCE=.\AGLayer.h
# End Source File
# Begin Source File

SOURCE=.\AGMatrix.h
# End Source File
# Begin Source File

SOURCE=.\AGPage.h
# End Source File
# Begin Source File
```

```
SOURCE=.\AGSym.h
# End Source File
# Begin Source File

SOURCE=.\AGText.h
# End Source File
# Begin Source File

SOURCE=.\Bsc2.h
# End Source File
# Begin Source File

SOURCE=.\CtlPanel.h
# End Source File
# Begin Source File

SOURCE=.\Ctp.h
# End Source File
# Begin Source File

SOURCE=.\dblside.h
# End Source File
# Begin Source File

SOURCE=.\Font.h
# End Source File
# Begin Source File

SOURCE=.\propsht.h
# End Source File
# Begin Source File

SOURCE=.\Resource.h
# End Source File
# Begin Source File

SOURCE=.\StdAfx.h
# End Source File
# Begin Source File

SOURCE=.\version.h
# End Source File
# Begin Source File

SOURCE=.\WaitDlg.h
# End Source File
# End Group
# Begin Group "Resource Files"

# PROP Default_Filter "ico;cur;bmp;dlg;rc2;rct;bin;cnt;rtf;gif;jpg
;jpeg;jpe"
# Begin Source File

SOURCE=.\Res\1up.bmp
# End Source File
# Begin Source File

SOURCE=.\Res\1up2down.bmp
# End Source File
# Begin Source File

SOURCE=.\Res\2down.bmp
# End Source File
# Begin Source File

SOURCE=.\Res\2up.bmp
# End Source File
# Begin Source File

SOURCE=.\Res\3up.bmp
# End Source File
# Begin Source File

SOURCE=.\Res\AGLogo.agi
# End Source File
# Begin Source File

SOURCE=".\Res\C&PLogo.agi"
# End Source File
# Begin Source File

SOURCE=.\Res\Cacfc___.ttz
# End Source File
# Begin Source File

SOURCE=.\Ctp.rgs
# End Source File
# Begin Source File

SOURCE=.\Res\Version.rc2
# End Source File
# End Group
# End Target
# End Project
```

```
#ifndef __AGTEXT_H_
#define __AGTEXT_H_

#include "AGDC.h"
#include "AGDoc.h"
#include "sctypes.h"
#include "scapptyp.h"
#include "scpubobj.h"
#include "scselect.h"


//////////////////////////////////////////////////////////////////////////////
// CAGSpec class
//
class CAGSpec : public stSpec
{
public:
    CAGSpec();
    CAGSpec(const LOGFONT &Font, COLORREF Color,
            eTSJust HorzJust = eRagRight, int nAboveParaLeading = 0,
            int nBelowParaLeading = 0, int nLineLeading = 0);
    CAGSpec(const CAGSpec &Spec);

    void operator = (const CAGSpec &Spec);
    int operator == (const CAGSpec &Spec) const;
    int operator != (const CAGSpec &Spec) const     { return (! operator==(Spec)); }

    void Read(CAGDocIO *pInput);
    void Write(CAGDocIO *pOutput) const;

public:
    UFont       m_Font;
    COLORREF    m_Color;
    eTSJust     m_HorzJust;
    int         m_nAboveParaLeading;
    int         m_nBelowParaLeading;
    int         m_nLineLeading;
};


//////////////////////////////////////////////////////////////////////////////
// CAGText class
//
class CAGText
{
public:
    CAGText();
    ~CAGText();
    void BlinkCursor();
    void Create(const RECT &DestRect);
    void DrawColumn(CAGDC &dc);
    void DrawSelection(CAGDC &dc);
    void Edit(CAGDC *pdc, int x, int y, bool bClick = false);
    void EndEdit();
    scColumn *GetColumn()                       { return (m_pColumn); }
    CAGDC *GetDC() const                        { return (m_pdc); }
    const RECT &GetDestRect () const            { return (m_DestRect); }
    CAGDocIO *GetDocIO() const                  { return (m_pDocIO); }
    void GetFonts(LOGFONTARRAY &lfArray);
    scSelection *GetSelection() const           { return (m_pSelection); }
    void GetSelParaTSList(scTypeSpecList &tsList) const;
    void GetSelTSList(scTypeSpecList &tsList) const;
    const scTypeSpecList &GetTSList() const     { return (m_TSList); }
    eVertJust GetVertJust() const;
    scTypeSpecList *GetWriteTSList() const      { return (m_pWriteTSList); }
    bool IsEditing() const                      { return (m_pSelection != NULL); }
    bool IsEmpty() const;
    bool IsLButtonDown() const                  { return (m_bLButtonDown); }
    void OnChar(UINT nChar);
    void OnKeyDown(UINT nChar);
    void OnKeyUp(UINT nChar);
    void OnLButtonDblClk(POINT Point);
    void OnLButtonDown(POINT Point, bool bShift);
    void OnLButtonUp(POINT Point);
    void OnMouseMove(POINT Point);
    void ReadColumn(CAGDocIO *pInput);
    void SetDC(CAGDC *pdc)                        { m_pdc = pdc; }

    void SetColor(COLORREF Color);
    void SetFont(const LOGFONT &Font);
    void SetHorzJust(eTSJust HorzJust);
    void SetNextSpec(const CAGSpec &Spec);
    void SetPtSize(int nPtSize);
    void SetText(const char *pText, int nChars, int nSpecs,
                const CAGSpec *pAGSpecs, const int *pSpecOffsets);

    void SetTypeface(const LOGFONT &Font);
    void SetUnderline(bool bUnderline);
    void SetVertJust(eVertJust VertJust);

    void ShowSelection(bool bShow);
    void WriteColumn(CAGDocIO *pOutput);

protected:
    void ComputeRedisplay(scRedispList &colRedisp);
    void ExtendSelection(eSelectMove movement);
    void MoveSelection(eSelectMove movement);
    void UpdateSelection();


protected:
    CAGDC           *m_pdc;
    RECT            m_DestRect;
    scColumn        *m_pColumn;
    scSelection     *m_pSelection;
    bool            m_bSelection;
    bool            m_bOverstrike;
    bool            m_bShiftKeyDown;
    bool            m_bControlKeyDown;
    bool            m_bLButtonDown;
    CAGSpec         *m_pNextSpec;
    scSelection     m_SelNextSpec;
    CAGDocIO        *m_pDocIO;
    scTypeSpecList  m_TSList;
    scTypeSpecList  *m_pWriteTSList;
    scMuPoint       m_LastPt;
};

#endif //__AGTEXT_H_
```

```
//============================================================================//
//============================================================================//
#include "stdafx.h"
#include "AGText.h"

#include "scappint.h"
#include "sccallbk.h"
#include "scstyle.h"
#include "scselect.h"
#include "scapptex.h"
#include "sccolumn.h"
#include "scparagr.h"

#ifdef _AFX
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
#endif

static long ReadFunc (APPCtxPtr pText, void *ptr, long size);
static long WriteFunc (APPCtxPtr pText, void *ptr, long size);


//----------------------------------------------------------------------------//
//----------------------------------------------------------------------------//
CAGSpec::CAGSpec()
{
    memset(&m_Font, 0, sizeof(m_Font));
    m_Color = 0;
    m_HorzJust = eRagRight;
    m_nAboveParaLeading = 0;
    m_nBelowParaLeading = 0;
    m_nLineLeading = 0;


//----------------------------------------------------------------------------//
//----------------------------------------------------------------------------//
CAGSpec::CAGSpec(const LOGFONT &Font, COLORREF Color, eTSJust HorzJust,
                int nAboveParaLeading, int nBelowParaLeading, int nLineLeading)
{
    m_Font = Font;
    m_Color = Color;
    m_HorzJust = HorzJust;
    m_nAboveParaLeading = nAboveParaLeading;
    m_nBelowParaLeading = nBelowParaLeading;
    m_nLineLeading = nLineLeading;
}


//----------------------------------------------------------------------------//
//----------------------------------------------------------------------------//
CAGSpec::CAGSpec(const CAGSpec &Spec)
{
    m_Font = Spec.m_Font;
    m_Color = Spec.m_Color;
    m_HorzJust = Spec.m_HorzJust;
    m_nAboveParaLeading = Spec.m_nAboveParaLeading;
    m_nBelowParaLeading = Spec.m_nBelowParaLeading;
    m_nLineLeading = Spec.m_nLineLeading;
}


//----------------------------------------------------------------------------//
//----------------------------------------------------------------------------//
void CAGSpec::operator=(const CAGSpec &Spec)
{
    m_Font = Spec.m_Font;
    m_Color = Spec.m_Color;
    m_HorzJust = Spec.m_HorzJust;
    m_nAboveParaLeading = Spec.m_nAboveParaLeading;
    m_nBelowParaLeading = Spec.m_nBelowParaLeading;
    m_nLineLeading = Spec.m_nLineLeading;
}


//------------------------------------------------------------------------------//
//------------------------------------------------------------------------------//
int CAGSpec::operator==(const CAGSpec &Spec) const
{
    return (m_Font == Spec.m_Font &&
            m_Color == Spec.m_Color &&
            m_HorzJust == Spec.m_HorzJust &&
            m_nAboveParaLeading == Spec.m_nAboveParaLeading &&
            m_nBelowParaLeading == Spec.m_nBelowParaLeading &&
            m_nLineLeading == Spec.m_nLineLeading);
}


//------------------------------------------------------------------------------//
//------------------------------------------------------------------------------//
void CAGSpec::Read(CAGDocIO *pInput)
{
    pInput->Read(&m_Font, sizeof(m_Font));
    pInput->Read(&m_Color, sizeof(m_Color));
    pInput->Read(&m_HorzJust, sizeof(m_HorzJust));
    pInput->Read(&m_nAboveParaLeading, sizeof(m_nAboveParaLeading));
    pInput->Read(&m_nBelowParaLeading, sizeof(m_nBelowParaLeading));
    pInput->Read(&m_nLineLeading, sizeof(m_nLineLeading));
}


//------------------------------------------------------------------------------//
//------------------------------------------------------------------------------//
void CAGSpec::Write(CAGDocIO *pOutput) const
{
    pOutput->Write(&m_Font, sizeof(m_Font));
    pOutput->Write(&m_Color, sizeof(m_Color));
    pOutput->Write(&m_HorzJust, sizeof(m_HorzJust));
    pOutput->Write(&m_nAboveParaLeading, sizeof(m_nAboveParaLeading));
    pOutput->Write(&m_nBelowParaLeading, sizeof(m_nBelowParaLeading));
    pOutput->Write(&m_nLineLeading, sizeof(m_nLineLeading));
}


//------------------------------------------------------------------------------//
//------------------------------------------------------------------------------//
CAGText::CAGText()
{
    ::SetRect(&m_DestRect, 0, 0, 0, 0);
    m_pColumn = NULL;
    m_pSelection = NULL;
    m_pdc = NULL;
    m_bSelection = false;
    m_bShiftKeyDown = false;
    m_bControlKeyDown = false;
    m_bLButtonDown = false;
    m_bOverstrike = false;
    m_pNextSpec = NULL;
    m_pWriteTSList = NULL;
}


//------------------------------------------------------------------------------//
//------------------------------------------------------------------------------//
CAGText::~CAGText()
{
    SCCOL_Delete(m_pColumn, 0);

    if (m_pNextSpec)
        delete m_pNextSpec;
}
```

```
//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CAGText::BlinkCursor()
{
    if (m_pSelection && m_pSelection->IsSliverCursor())
    {
        if (!m_bLButtonDown)
            ShowSelection(!m_bSelection);
    }
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CAGText::ComputeRedisplay(scRedispList &colRedisp)
{
    scColRedisplay colRect;

    for (int limit = 0; limit < colRedisp.GetNumItems(); limit++ )
    {
        colRedisp.GetDataAt(limit, (ElementPtr)&colRect);

        if (colRect.fHasDamage)
        {
            scXRect dRect(colRect.fDamageRect);

            RECT r = { m_DestRect.left + dRect.x1, m_DestRect.top + dRect.y1,
                        m_DestRect.left + dRect.x2, m_DestRect.top + dRect.y2 };

            POINT Pts[4];
            Pts[0].x = Pts[1].x = r.left;
            Pts[2].x = Pts[3].x = r.right;
            Pts[0].y = Pts[3].y = r.top;
            Pts[1].y = Pts[2].y = r.bottom;
            m_pdc->LPtoDP(Pts, 4);

            r.left = r.top = LONG_MAX;
            r.right = r.bottom = LONG_MIN;
            for (int i = 0; i < 4; i++)
            {
                if (Pts[i].x < r.left)
                    r.left = Pts[i].x;
                if (Pts[i].x > r.right)
                    r.right = Pts[i].x;
                if (Pts[i].y < r.top)
                    r.top = Pts[i].y;
                if (Pts[i].y > r.bottom)
                    r.bottom = Pts[i].y;
            }

            ::InvalidateRect(m_pdc->GetWnd(), &r, false);
        }

        if (colRect.fHasRepaint)
        {
            scXRect rRect(colRect.fRepaintRect);

            RECT r = { m_DestRect.left + rRect.x1, m_DestRect.top + rRect.y1,
                        m_DestRect.left + rRect.x2, m_DestRect.top + rRect.y2 };

            POINT Pts[4];
            Pts[0].x = Pts[1].x = r.left;
            Pts[2].x = Pts[3].x = r.right;
            Pts[0].y = Pts[3].y = r.top;
            Pts[1].y = Pts[2].y = r.bottom;
            m_pdc->LPtoDP(Pts, 4);

            r.left = r.top = LONG_MAX;
            r.right = r.bottom = LONG_MIN;
            for (int i = 0; i < 4; i++)
            {
                if (Pts[i].x < r.left)
                    r.left = Pts[i].x;
                if (Pts[i].x > r.right)
                    r.right = Pts[i].x;
                if (Pts[i].y < r.top)
                    r.top = Pts[i].y;
                if (Pts[i].y > r.bottom)
                    r.bottom = Pts[i].y;
            }

            ::InvalidateRect(m_pdc->GetWnd(), &r, false);
        }

        if (colRect.fImmediateRedisplay)
        {
            scXRect iRect(colRect.fImmediateArea.fImmediateRect);
            SCCOL_UpdateLine(colRect.fColumnID, colRect.fImmediateArea, this);
        }
    }
}


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
void CAGText::Create(const RECT &DestRect)
{
    m_DestRect = DestRect;

    SCCOL_New(this, m_pColumn, WIDTH(m_DestRect), HEIGHT(m_DestRect));
    SCFS_Recompose(m_pColumn, 0);
}


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
void CAGText::DrawColumn(CAGDC &dc)
{
    CAGDC *pSaveDC = m_pdc;
    m_pdc = &dc;

    scXRect ClipRect(0, 0, WIDTH(m_DestRect), HEIGHT(m_DestRect));
    SCCOL_Update(m_pColumn, ClipRect, this);

    m_pdc = pSaveDC;
}


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
void CAGText::DrawSelection(CAGDC &dc)
{
    if (m_pSelection && m_bSelection)
    {
        CAGDC *pSaveDC = m_pdc;
        m_pdc = &dc;

        SCSEL_Hilite(m_pSelection, APPDrawRect);

        m_pdc = pSaveDC;
    }
}


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
void CAGText::Edit(CAGDC *pdc, int x, int y, bool bClick)
{
    m_pdc = pdc;
    ShowSelection(false);

    if (IsEmpty())
    {
        TypeSpec ts(m_pNextSpec);
        SCCOL_InitialSelect(m_pColumn, ts, m_pSelection);
        m_pNextSpec = NULL;
```

```
            ShowSelection(true);
        }
        else
        {
            scMuPoint pt(x - m_DestRect.left, y - m_DestRect.top);
            SCCOL_StartSelect(m_pColumn, pt, APPDrawRect, this, m_pSelection);
            m_bSelection = true;
        }
        m_bShiftKeyDown = false;
        m_bControlKeyDown = false;
        m_bOverstrike = false;
        m_bLButtonDown = bClick;
        m_LastPt.Invalidate();
}


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
void CAGText::EndEdit()
{
        ShowSelection(false);
        m_pdc = NULL;
        m_pSelection = NULL;
        m_bShiftKeyDown = false;
        m_bControlKeyDown = false;
        m_bLButtonDown = false;
        m_bOverstrike = false;

        if (m_pNextSpec)
        {
            m_pNextSpec = NULL;
            delete m_pNextSpec;
        }
}


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
void CAGText::ExtendSelection(eSelectMove movement)
{
        ShowSelection(false);
        SCSEL_Extend(m_pSelection, movement);
        UpdateSelection();
        ShowSelection(true);
}


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
void CAGText::GetFonts(LOGFONTARRAY &lfArray)
{
        scTypeSpecList TSList;
        SCCOL_TSList(m_pColumn, TSList);

        int nNumItems = TSList.NumItems();

        for (int i = 0; i < nNumItems; i++)
        {
            LOGFONT lf = ((CAGSpec *)TSList[i].ptr())->m_Font;

            int nFonts = lfArray.size();

            for (int j = 0; j < nFonts; j++)
            {
                if (lstrcmp(lf.lfFaceName, lfArray[j].lfFaceName) == 0 &&
                    lf.lfWeight == lfArray[j].lfWeight &&
                    (lf.lfItalic != 0) == (lfArray[j].lfItalic != 0))
                {
                    break;
                }
            }
            if (j >= nFonts)
                lfArray.push_back(lf);
```

```
    }
}


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
void CAGText::GetSelParaTSList(scTypeSpecList &tsList) const
{
    SCSEL_PARATSList(m_pSelection, tsList);
}


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
void CAGText::GetSelTSList(scTypeSpecList &tsList) const
{
    if (m_pNextSpec)
    {
        TypeSpec ts(new CAGSpec(*m_pNextSpec));
        tsList.Insert(ts);
    }
    else if (m_pSelection)
        SCSEL_TSList(m_pSelection, tsList);
}


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
eVertJust CAGText::GetVertJust() const
{
    if (m_pColumn)
        return (m_pColumn->GetVertJust());
    else
        return (eVertTop);
}


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
bool CAGText::IsEmpty() const
{
    long lChCount = 0;

    if (m_pColumn)
    {
        scStream *pStream = NULL;
        SCCOL_GetStream(m_pColumn, pStream);

        if (pStream)
            SCSTR_ChCount(pStream, lChCount);
    }

    return (lChCount == 0);
}


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
void CAGText::MoveSelection(eSelectMove movement)
{
    ShowSelection(false);
    SCSEL_Move(m_pSelection, movement);
    UpdateSelection();
    ShowSelection(true);
}


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
void CAGText::OnChar(UINT nChar)
{
    if (m_pSelection && ! m_bLButtonDown)
    {
        scRedispList    colRedisp;
        scKeyRecord     keyRec;

        keyRec.type() = m_bOverstrike ? scKeyRecord::overstrike : scKeyRecord::insert;
        keyRec.keycode() = CMappToCT(nChar);
        if (m_pNextSpec)
        {
            TypeSpec Spec(m_pNextSpec);
            keyRec.spec() = Spec;
            m_pNextSpec = NULL;
        }
        else
        {
            TypeSpec NullSpec;
            keyRec.spec() = NullSpec;
        }

        if (keyRec.keycode() == scParaSplit)
            keyRec.keycode() = scHardReturn;
        else if (keyRec.keycode() == scHardReturn)
            keyRec.keycode() = scParaSplit;

        ShowSelection(false);
        SCSEL_InsertKeyRecords(m_pSelection, 1, &keyRec, &colRedisp);
        ComputeRedisplay(colRedisp);
        ShowSelection(true);
    }
}

//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CAGText::OnKeyDown(UINT nChar)
{
    if (m_pSelection && !m_bLButtonDown)
    {
        switch (nChar)
        {
            case VK_CONTROL:
                m_bControlKeyDown = true;
                break;

            case VK_SHIFT:
                m_bShiftKeyDown = true;
                break;

            case VK_INSERT:
                m_bOverstrike = !m_bOverstrike;
                break;

            case VK_DELETE:
            {
                scRedispList    colRedisp;
                scKeyRecord     keyRec;

                keyRec.keycode() = (UCS2)scForwardDelete;
                ShowSelection(false);
                SCSEL_InsertKeyRecords(m_pSelection, 1, &keyRec, &colRedisp);
                ComputeRedisplay(colRedisp);
                ShowSelection(true);
                break;
            }

            case VK_LEFT:
                if (m_bShiftKeyDown)
                    ExtendSelection(m_bControlKeyDown ? ePrevWord : ePrevChar);
                else
                    MoveSelection(m_bControlKeyDown ? ePrevWord : ePrevChar);
                break;

            case VK_RIGHT:
                if (m_bShiftKeyDown)
                    ExtendSelection(m_bControlKeyDown ? eNextWord : eNextChar);
                else
                    MoveSelection(m_bControlKeyDown ? eNextWord : eNextChar);
                break;

            case VK_UP:
                if (m_bShiftKeyDown)
                    ExtendSelection(ePrevLine);
                else
                    MoveSelection(ePrevLine);
                break;

            case VK_DOWN:
                if (m_bShiftKeyDown)
                    ExtendSelection(eNextLine);
                else
                    MoveSelection(eNextLine);
                break;

            case VK_HOME:
                if (m_bShiftKeyDown)
                    ExtendSelection(m_bControlKeyDown ? eBeginColumn : eStartLine);
                else
                    MoveSelection(m_bControlKeyDown ? eBeginColumn : eStartLine);
                break;

            case VK_END:
                if (m_bShiftKeyDown)
                    ExtendSelection(m_bControlKeyDown ? eEndColumn : eEndLine);
                else
                    MoveSelection(m_bControlKeyDown ? eEndColumn : eEndLine);
                break;

            case VK_PRIOR:
                if (m_bShiftKeyDown)
                    ExtendSelection(eBeginColumn);
                else
                    MoveSelection(eBeginColumn);
                break;

            case VK_NEXT:
                if (m_bShiftKeyDown)
                    ExtendSelection(eEndColumn);
                else
                    MoveSelection(eEndColumn);
                break;

            default:
                break;
        }

        if (m_pNextSpec && !(*m_pSelection == m_SelNextSpec))
        {
            delete m_pNextSpec;
            m_pNextSpec = NULL;
        }
    }
}

//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CAGText::OnKeyUp(UINT nChar)
{
    switch (nChar)
    {
        case VK_CONTROL:
            m_bControlKeyDown = false;
            break;

        case VK_SHIFT:
            m_bShiftKeyDown = false;
            break;
        default:
            break;
    }
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CAGText::OnLButtonDblClk(POINT Point)
{
    if (m_pSelection)
    {
        scMuPoint muPt(Point.x - m_DestRect.left, Point.y - m_DestRect.top);
        ShowSelection(false);
        SCCOL_SelectSpecial(m_pColumn, muPt, eWordSelect, m_pSelection);
        ShowSelection(true);
    }
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CAGText::OnLButtonDown(POINT Point, bool bShift)
{
    if (m_pSelection)
    {
        ShowSelection(false);
        scMuPoint muPt(Point.x - m_DestRect.left, Point.y - m_DestRect.top);
        SCCOL_StartSelect(m_pColumn, muPt, APPDrawRect, this, m_pSelection);
        m_bSelection = true;
        m_bLButtonDown = true;
        m_LastPt.Invalidate();

        if (m_pNextSpec && !(*m_pSelection == m_SelNextSpec))
        {
            delete m_pNextSpec;
            m_pNextSpec = NULL;
        }
    }
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CAGText::OnLButtonUp(POINT Point)
{
    if (m_pSelection)
    {
        m_bLButtonDown = false;
        if (m_pSelection && !m_pSelection->IsSliverCursor())
            m_bSelection = TRUE;

        if (m_pNextSpec && !(*m_pSelection == m_SelNextSpec))
        {
            delete m_pNextSpec;
            m_pNextSpec = NULL;
        }
    }
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CAGText::OnMouseMove(POINT Point)
{
    if (m_pSelection && m_bLButtonDown)
    {
        scMuPoint muPt(Point.x - m_DestRect.left, Point.y - m_DestRect.top);
        if (muPt.x < 0)
            muPt.x = 0;
        if (muPt.x > WIDTH(m_DestRect))
            muPt.x = WIDTH(m_DestRect);
        if (muPt.y < 0)
```

```
                muPt.y = 0;
            if (muPt.y > HEIGHT(m_DestRect))
                muPt.y = HEIGHT(m_DestRect);

            if (m_LastPt != muPt)
            {
                SCCOL_ExtendSelect(m_pColumn, muPt, APPDrawRect, this, m_pSelection);
                m_LastPt = muPt;
            }
        }
}

//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CAGText::ReadColumn(CAGDocIO *pInput)
{
    if (m_pColumn)
        SCCOL_Delete(m_pColumn, 0);
    if (m_pNextSpec)
        delete m_pNextSpec;

    m_TSList.RemoveAll();

    m_pColumn = NULL;
    m_pSelection = NULL;
    m_pdc = NULL;
    m_bShiftKeyDown = false;
    m_bControlKeyDown = false;
    m_bLButtonDown = false;
    m_bOverstrike = false;
    m_pNextSpec = NULL;

    m_pDocIO = pInput;
    pInput->Read(&m_DestRect, sizeof(m_DestRect));

    int nNumItems = 0;
    pInput->Read(&nNumItems, sizeof(nNumItems));

    for (int i = 0; i < nNumItems; i++)
    {
        CAGSpec *pAGSpec = new CAGSpec();
        pAGSpec->Read(pInput);

        TypeSpec ts(pAGSpec);
        m_TSList.Set(i, ts);
    }

    scSet *enumTable;
    SCSET_InitRead(enumTable, 100);
    SCCOL_Read(this, m_pColumn, enumTable, this, ReadFunc);
    SCOBJ_PtrRestore((scTBObj *)m_pColumn, enumTable);
    SCSET_FiniRead(enumTable, 0);

    m_pDocIO = NULL;
}

//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CAGText::SetColor(COLORREF Color)
{
    if (m_pSelection)
    {
        if (m_pSelection->IsSliverCursor())
        {
            if (m_pNextSpec == NULL)
            {
                TypeSpec ts = m_pSelection->fMark.fPara->SpecAtOffset(m_pSelection->fMark.fOffset);
                m_pNextSpec = new CAGSpec(*((CAGSpec *)ts.ptr()));
                m_SelNextSpec = *m_pSelection;
            }
            m_pNextSpec->m_Color = Color;
        }
        else
        {
            ShowSelection(false);

            scStream *fStream;
            SCCOL_GetStream(m_pColumn, fStream);
            scSpecLocList redocsl(fStream);
            SCSEL_CHTSList(m_pSelection, redocsl);

            for (int i = 0; i < redocsl.NumItems(); i++)
            {
                if (redocsl[i].spec().ptr())
                {
                    CAGSpec *pNewSpec = new CAGSpec(*((CAGSpec *)redocsl[i].spec().ptr()));
                    pNewSpec->m_Color = Color;
                    redocsl[i].spec() = pNewSpec;
                }
            }
            scRedispList colRedisp;
            SCSTR_CHTSListSet(fStream, redocsl, &colRedisp);
            ComputeRedisplay(colRedisp);
            ShowSelection(true);
        }
    }
}

//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CAGText::SetFont (const LOGFONT &Font)
{

}

//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CAGText::SetHorzJust(eTSJust HorzJust)
{
    scStream *fStream;
    SCCOL_GetStream(m_pColumn, fStream);
    scSpecLocList csl(fStream);

    SCSEL_PARATSList(m_pSelection, csl);

    for (int i = 0; i < csl.NumItems(); i++)
    {
        if (csl[i].spec().ptr())
        {
            CAGSpec *pNewSpec = new CAGSpec(*((CAGSpec *)csl[i].spec().ptr()));
            pNewSpec->m_HorzJust = HorzJust;
            csl[i].spec() = pNewSpec;
        }
    }
    scRedispList colRedisp;
    SCSTR_PARATSListSet(fStream, csl, &colRedisp);
    ComputeRedisplay(colRedisp);
}

//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CAGText::SetNextSpec(const CAGSpec &Spec)
{
    if (m_pNextSpec)
        delete m_pNextSpec;

    m_pNextSpec = new CAGSpec(Spec);
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
```

```
void CAGText::SetPtSize(int nPtSize)
{
    if (m_pSelection)
    {
        if (m_pSelection->IsSliverCursor())
        {
            if (m_pNextSpec == NULL)
            {
                TypeSpec ts = m_pSelection->fMark.fPara->SpecAtOffset(m_pSelection->fMark.fOffset);
                m_pNextSpec = new CAGSpec(*((CAGSpec *)ts.ptr()));
                m_SelNextSpec = *m_pSelection;
            }
            m_pNextSpec->m_Font.lfHeight = -nPtSize * APP_RESOLUTION / 72;
        }
        else
        {
            ShowSelection(false);

            scStream *fStream;
            SCCOL_GetStream(m_pColumn, fStream);
            scSpecLocList redocsl(fStream);
            SCSEL_CHTSList(m_pSelection, redocsl);

            for (int i = 0; i < redocsl.NumItems(); i++)
            {
                if (redocsl[i].spec().ptr())
                {
                    CAGSpec *pNewSpec = new CAGSpec(*((CAGSpec *)redocsl[i].spec().ptr()));
                    pNewSpec->m_Font.lfHeight = -nPtSize * APP_RESOLUTION / 72;
                    redocsl[i].spec() = pNewSpec;
                }
            }
            scRedispList colRedisp;
            SCSTR_CHTSListSet(fStream, redocsl, &colRedisp);
            ComputeRedisplay(colRedisp);
            ShowSelection(true);
        }
    }
}

//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CAGText::SetText(const char *pText, int nChars, int nSpecs,
                        const CAGSpec *pAGSpecs, const int *pSpecOffsets)
{
    bool bStartNewPara = true;

    stTextImportExport &AppText = stTextImportExport::MakeTextImportExport();
    CAGSpec *pAGSpec = NULL;

    for (int i = 0; i < nSpecs; i++)
    {
        int nLen;
        if ((i + 1) < nSpecs)
            nLen = pSpecOffsets[i + 1] - pSpecOffsets[i];
        else
            nLen = nChars - pSpecOffsets[i];

        if (nLen)
        {
            pAGSpec = new CAGSpec(pAGSpecs[i]);
            TypeSpec ts(pAGSpec);

            if (bStartNewPara)
            {
                AppText.StartPara(ts);
                bStartNewPara = false;
            }

            for (int j = 0; j < nLen; j++)
            {
                if (pText[j] == 0x12)
                {
                    if (bStartNewPara)
                        AppText.StartPara(ts);

                    if (j > 0)
                        AppText.PutString((const uchar *)pText, j, ts);

                    bStartNewPara = true;
                    nLen -= (j + 1);
                    pText += (j + 1);
                    j = -1;
                }
            }

            if (nLen)
            {
                if (bStartNewPara)
                {
                    AppText.StartPara(ts);
                    bStartNewPara = false;
                }

                AppText.PutString((const uchar *)pText, nLen, ts);
                pText += nLen;
            }
        }
    }

    AppText.reset();
    SCFS_PasteAPPText(m_pColumn, AppText, 0);
}

//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CAGText::SetTypeface(const LOGFONT &Font)
{
    if (m_pSelection)
    {
        if (m_pSelection->IsSliverCursor())
        {
            if (m_pNextSpec == NULL)
            {
                TypeSpec ts = m_pSelection->fMark.fPara->SpecAtOffset(m_pSelection->fMark.fOffset);
                m_pNextSpec = new CAGSpec(*((CAGSpec *)ts.ptr()));
                m_SelNextSpec = *m_pSelection;
            }
            lstrcpy(m_pNextSpec->m_Font.lfFaceName, Font.lfFaceName);
            m_pNextSpec->m_Font.lfWeight = Font.lfWeight;
            m_pNextSpec->m_Font.lfItalic = (Font.lfItalic != 0);
            m_pNextSpec->m_Font.lfCharSet = Font.lfCharSet;
            m_pNextSpec->m_Font.lfOutPrecision = Font.lfOutPrecision;
            m_pNextSpec->m_Font.lfClipPrecision = Font.lfClipPrecision;
            m_pNextSpec->m_Font.lfQuality = Font.lfQuality;
            m_pNextSpec->m_Font.lfPitchAndFamily = Font.lfPitchAndFamily;
        }
        else
        {
            ShowSelection(false);

            scStream *fStream;
            SCCOL_GetStream(m_pColumn, fStream);
            scSpecLocList redocsl(fStream);
            SCSEL_CHTSList(m_pSelection, redocsl);

            for (int i = 0; i < redocsl.NumItems(); i++ )
            {
                if (redocsl[i].spec().ptr())
                {
                    CAGSpec *pNewSpec = new CAGSpec(*((CAGSpec *)redocsl[i].spec().ptr()));
                    lstrcpy(pNewSpec->m_Font.lfFaceName, Font.lfFaceName);
                    pNewSpec->m_Font.lfWeight = Font.lfWeight;
                    pNewSpec->m_Font.lfItalic = (Font.lfItalic != 0);
                    pNewSpec->m_Font.lfCharSet = Font.lfCharSet;
                    pNewSpec->m_Font.lfOutPrecision = Font.lfOutPrecision;
                    pNewSpec->m_Font.lfClipPrecision = Font.lfClipPrecision;
                    pNewSpec->m_Font.lfQuality = Font.lfQuality;
                    pNewSpec->m_Font.lfPitchAndFamily = Font.lfPitchAndFamily;
                    redocsl[i].spec() = pNewSpec;
                }
            }
            scRedispList colRedisp;
            SCSTR_CHTSListSet(fStream, redocsl, &colRedisp);
            ComputeRedisplay(colRedisp);
            ShowSelection(true);
        }
    }
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CAGText::SetUnderline(bool bUnderline)
{
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CAGText::SetVertJust(eVertJust VertJust)
{
    ShowSelection(false);

    SCCOL_FlowJustify(m_pColumn, VertJust);
    SCFS_Recompose(m_pColumn, 0);

    if (m_pSelection)
    {
        RECT r = m_DestRect;
        m_pdc->LPtoDP(&r);
        ::InvalidateRect(m_pdc->GetWnd(), &r, false);
    }

    ShowSelection(true);
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CAGText::ShowSelection(bool bShow)
{
    if (m_pSelection)
    {
        if (bShow != m_bSelection)
        {
            SCSEL_Hilite(m_pSelection, APPDrawRect);
            m_bSelection = bShow;
        }
    }
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CAGText::UpdateSelection()
{
    scStream        *streamID;
    scStreamLocation Mark, Point;

    SCSEL_Decompose2(m_pSelection, Mark, Point);
    SCCOL_GetStream(m_pColumn, streamID);
    SCSEL_Restore(streamID, Mark, Point, m_pSelection, false);
}


//--------------------------------------------------------------------------//
```

```
//---------------------------------------------------------------------------//
void CAGText::WriteColumn(CAGDocIO *pOutput)
{
    if (m_pColumn)
    {
        m_pDocIO = pOutput;
        pOutput->Write(&m_DestRect, sizeof(m_DestRect));

        scStream *pStream = NULL;
        SCCOL_GetStream(m_pColumn, pStream);

        SCTSL_Alloc(m_pWriteTSList);
        SCSTR_ParaTSList(pStream, *m_pWriteTSList);
        SCSTR_TSList(pStream, *m_pWriteTSList);

        int nNumItems = m_pWriteTSList->NumItems();
        pOutput->Write(&nNumItems, sizeof(nNumItems));

        for (int i = 0; i < nNumItems; i++)
        {
            ((CAGSpec *)((*m_pWriteTSList)[i].ptr()))->Write(pOutput);
        }

        long enumerate = 0;
        SCTB_ZeroEnumeration();
        SCOBJ_Enumerate((scTBObj*)m_pColumn, enumerate);
        SCCOL_Write(m_pColumn, this, WriteFunc);

        SCTSL_Delete(m_pWriteTSList);
        m_pDocIO = NULL;
    }
}

//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
static long ReadFunc(APPCtxPtr pText, void *ptr, long size)
{
    CAGDocIO *pDocIO = pText->GetDocIO();
    if (pDocIO)
        pDocIO->Read(ptr, size);
    return (size);
}

//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
static long WriteFunc(APPCtxPtr pText, void *ptr, long size)
{
    CAGDocIO *pDocIO = pText->GetDocIO();
    if (pDocIO)
        pDocIO->Write(ptr, size);
    return (size);
}

//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
scTicks SCTickCount()
{
    return GetTickCount();
}

//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
scTicks APPEventAvail(scProcType proctype)
{
    switch (proctype)
    {
        case scDrawProc:
            return 1000;
        case scReformatProc:
/*
            if (IsKeyStrokes())
                return 0;
*/
            return 20;
    }
    return 0;
}

//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
void *APPDiskIDToPointer(APPCtxPtr pText, long index, stDiskidClass objclass)
{
    switch (objclass)
    {
        case diskidColumn:
            return (pText);

        case diskidTypespec:
        {
            if (index <= 0)
                return (NULL);

            const scTypeSpecList &TSList = pText->GetTSList();
            CAGSpec *pAGSpec = (CAGSpec *)(TSList[index - 1].ptr());
            return (pAGSpec);
        }

        default:
            break;
    }

    return (void*)-1;


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
long APPPointerToDiskID(APPCtxPtr pText, void *obj, stDiskidClass objclass)
{
    switch (objclass)
    {
        case diskidTypespec:
        {
            if (obj == NULL)
                return (0);

            scTypeSpecList *pWriteTSList = pText->GetWriteTSList();
            int nNumItems = pWriteTSList->NumItems();
            int nItem = -1;
            for (int i = 0; i < nNumItems; i++)
            {
                if ((*pWriteTSList)[i].ptr() == obj)
                {
                    nItem = i + 1;
                    break;
                }
            }
            return (nItem);
        }

        default:
            break;
    }

    return (-1);
}

//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
```

```
Bool APPRecomposeColumn(APPColumn)
{
    return true;
}


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
void APPDrawStartLine(APPDrwCtx pText, MicroPoint xOrg, MicroPoint yOrg,
                        const scXRect &inkExtents)
{
}


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
void APPDrawEndLine(APPDrwCtx pText)
{
}


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
void APPDrawString(APPDrwCtx pText, const scGlyphArray *glyphArray, short num,
                    MicroPoint xOrg, MicroPoint yOrg, const scGlyphInfo &gi)
{
    TCHAR   *pString = new TCHAR [num];
    int     *pWidths = new int [num];

    for (int i = 0; i < num; i++, glyphArray++)
    {
        pString[i] = LOBYTE(glyphArray->fGlyphID);
        pWidths[i] = glyphArray->hEscapement;
    }

    CAGSpec *pSpec = (CAGSpec *)gi.typespec.ptr();
    CAGDC *pDC = pText->GetDC();

    pDC->SetFont(pSpec->m_Font);
    pDC->SetTextColor(pSpec->m_Color);

    RECT DestRect = pText->GetDestRect();
    pDC->ExtTextOut(DestRect.left + xOrg, DestRect.top + yOrg,
                    0, NULL, pString, num, pWidths);

    delete[] pString;
    delete[] pWidths;
}


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
void APPDrawRect(const scXRect &xrect, APPDrwCtx pText, Bool sliverCursor)
{
    CAGDC *pDC = pText->GetDC();
    if (pDC)
    {
        RECT DestRect = pText->GetDestRect();
        RECT r;

        r.left = max((DestRect.left + xrect.x1), DestRect.left);
        r.top = max((DestRect.top + xrect.y1), DestRect.top);
        r.right = min((DestRect.left + xrect.x2), DestRect.right);
        r.bottom = min((DestRect.top + xrect.y2), DestRect.bottom);

        if (WIDTH(r) >= 0 && HEIGHT(r) >= 0)
        {
            if (WIDTH(r) == 0)
            {
                POINT ptFrom = { r.left, r.top };
                POINT ptTo = { r.left, r.bottom };
                pDC->InvertLine(ptFrom, ptTo);
            }
            else
                pDC->InvertRect(r);
        }
    }
}


//------------------------------------------------------------------------//
//------------------------------------------------------------------------//
status APPDrawContext(APPColumn appID, const scColumn *colid, APPDrwCtx &pText)
{
    pText = appID;
    return scSuccess;
}


//------------------------------------------------------------------------//
//------------------------------------------------------------------------//
void APPDrawRule(const scMuPoint &start, const scMuPoint &end,
                    const scGlyphInfo &gi, APPDrwCtx pText)
{
}


//------------------------------------------------------------------------//
//------------------------------------------------------------------------//
RLU FIgetRLUEscapement(const scFontRender &fp, UCS2 ch)
{
    return (0);
}


//------------------------------------------------------------------------//
//------------------------------------------------------------------------//
RLU FIgetRLUKern(const scFontRender &fp, UCS2 ch1, UCS2 ch2)
{
    return (0);
}


//------------------------------------------------------------------------//
//------------------------------------------------------------------------//
scRLURect &FIgetRLUExtents(const scFontRender &fp, UCS2 ch, scRLURect &chBox)
{
    return (chBox);
}


//------------------------------------------------------------------------//
//------------------------------------------------------------------------//
void FIgetRLUFontExtents(const scFontRender &fp, RLU &capHite, RLU &xHite,
                            RLU &ascenderHite, RLU &descenderDepth, scRLURect &maxFont)
{
    capHite = 0;
    xHite = 0;
    ascenderHite = 0;
    descenderDepth = 0;

    maxFont.rluLeft = 0;
    maxFont.rluTop = 0;
    maxFont.rluRight = 0;
    maxFont.rluBottom = 0;

    CAGSpec *pSpec = (CAGSpec *)fp.ptr();
    CAGIC TextIC("DISPLAY");
    TextIC.SetTransformMode(false);
    TextIC.SetFont(pSpec->m_Font);

    TEXTMETRIC tm;
    TextIC.GetTextMetrics(&tm);

    capHite = (short)(scRoundMP((REAL)(tm.tmAscent - tm.tmInternalLeading) *
                scBaseRLUsystem / tm.tmHeight));
```

```
}


//-------------------------------------------------------------------------//
//-------------------------------------------------------------------------//
GlyphSize FIgetDEVEscapement(const scFontRender &fp, UCS2 ch)
{
    CAGSpec *pSpec = (CAGSpec *)fp.ptr();
    CAGIC TextIC("DISPLAY");
    TextIC.SetTransformMode(false);
    TextIC.SetFont(pSpec->m_Font);

    TCHAR temp = (TCHAR)ch;
    SIZE sizeExtent;
    TextIC.GetTextExtent(&temp, 1, &sizeExtent);

    return (sizeExtent.cx);
}


//-------------------------------------------------------------------------//
//-------------------------------------------------------------------------//
MicroPoint FIgetDEVEscapement(const scFontRender &, UCS2, MicroPoint suggestedWidth)
{
    return suggestedWidth;
}


//-------------------------------------------------------------------------//
//-------------------------------------------------------------------------//
GlyphSize FIgetDEVKern(const scFontRender &fp, UCS2 ch1, UCS2 ch2)
{
    return 0;
}


//-------------------------------------------------------------------------//
//-------------------------------------------------------------------------//
scXRect &FIgetDEVExtents(const scFontRender &fp, UCS2 ch, scXRect &chBox)
{
    return chBox;
}


//-------------------------------------------------------------------------//
//-------------------------------------------------------------------------//
void FIgetDEVFontExtents(const scFontRender &fp, MicroPoint &capHite, MicroPoint &xHite,
                         MicroPoint &ascenderHite, MicroPoint &descenderDepth,
                         scXRect &maxFont)
{
    CAGSpec *pSpec = (CAGSpec *)fp.ptr();
    CAGIC TextIC("DISPLAY");
    TextIC.SetTransformMode(false);
    TextIC.SetFont(pSpec->m_Font);

    TEXTMETRIC tm;
    TextIC.GetTextMetrics(&tm);
    int nMinAWidth = TextIC.GetMinAWidth(tm.tmFirstChar, tm.tmLastChar);

    capHite = tm.tmAscent;
    xHite = 0;                      // Not used
    ascenderHite = tm.tmAscent;
    descenderDepth = -tm.tmDescent;

    maxFont.x1 = nMinAWidth;
    maxFont.y1 = tm.tmAscent;
    maxFont.x2 = tm.tmMaxCharWidth;
    maxFont.y2 = -(tm.tmDescent + tm.tmExternalLeading);
}


//-------------------------------------------------------------------------//
//-------------------------------------------------------------------------//
```

```
Bool HYFWord(const UCS2 *word, short *hyfArray)
{
    return false;
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
status TSTabInfo(TypeSpec &ps, TypeSpec &ts,scTabInfo &tabInfo,
                MicroPoint xPos, MicroPoint, long)
{
    int nTabSize = one_inch / 4;
    tabInfo.xPosition = xPos + (nTabSize - (xPos % nTabSize));
    tabInfo.tabAlign = eTBLeftAlign;
    tabInfo.fillChar = ' ';
    tabInfo.fillCharAlign = eFCNormalAlign;

    return scSuccess;
}

//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
status TSfillCharInfo(TypeSpec &, UCS2 &, eFCAlignment &, MicroPoint,
                        MicroPoint, long)
{
    return scSuccess;
}

//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
status TSGetStyle(TypeSpec &ts, scStyle &st)
{
    CAGSpec *pSpec = (CAGSpec *)ts.ptr();
    if (pSpec)
    {
        CAGIC TextIC("DISPLAY");
        TextIC.SetTransformMode(false);
        TextIC.SetFont(pSpec->m_Font);

        TEXTMETRIC tm;
        TextIC.GetTextMetrics(&tm);

        SIZE sizeSpace;
        TextIC.GetTextExtent(" ", 1, &sizeSpace);

        scStyle style(pSpec->m_Font.lfFaceName, tm.tmHeight);
        style.SetRag(pSpec->m_HorzJust);

        if (pSpec->m_nLineLeading == 0)
        {
            style.SetLead(0);
            style.SetMaxLead(0);
        }
        else
        {
            style.SetLead(pSpec->m_nLineLeading - tm.tmHeight);
            style.SetMaxLead(pSpec->m_nLineLeading - tm.tmHeight);
        }

        if (pSpec->m_nAboveParaLeading == 0)
        {
            style.SetMaxSpaceAbove(0);
            style.SetSpaceAbove(0);
        }
        else
        {
            style.SetMaxSpaceAbove(pSpec->m_nAboveParaLeading - tm.tmHeight);
            style.SetSpaceAbove(pSpec->m_nAboveParaLeading - tm.tmHeight);
        }

        style.SetMaxSpaceBelow(0);
        style.SetSpaceBelow(0);
        style.SetAbsoluteLead(0);

        style.SetOptWord(sizeSpace.cx);
        style.SetMinWord(sizeSpace.cx);
        style.SetMaxWord(sizeSpace.cx);

        style.SmallCapCorrection();
        st = style;
    }
    else
    {
        scStyle style("Times New Roman", 15);
        st = style;
    }
    return scSuccess;
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
Bool TSdropCap(TypeSpec &pspec, TypeSpec &charspec, DCPosition &dcpos,
                UCS2 dropchar)
{
    return false;
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
MicroPoint CSfirstLinePosition(APPColumn pText, TypeSpec ts)
{
    CAGSpec *pSpec = (CAGSpec *)ts.ptr();
    CAGIC TextIC("DISPLAY");
    TextIC.SetTransformMode(false);
    TextIC.SetFont(pSpec->m_Font);

    TEXTMETRIC tm;
    TextIC.GetTextMetrics(&tm);

    if (pText->GetVertJust() == eVertCentered)
        return (tm.tmHeight + tm.tmExternalLeading);
    else
        return (tm.tmAscent);
}

//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
MicroPoint CSlastLinePosition(APPColumn pText, TypeSpec ts)
{
    if (pText->GetColumn()->GetVertJust() == eVertBottom)
    {
        CAGSpec *pSpec = (CAGSpec *)ts.ptr();
        CAGIC TextIC("DISPLAY");
        TextIC.SetTransformMode(false);
        TextIC.SetFont(pSpec->m_Font);

        TEXTMETRIC tm;
        TextIC.GetTextMetrics(&tm);

        return (tm.tmDescent + tm.tmExternalLeading);
    }
    return (0);
}


stUnivChar  ucnull[]        = { '\0' };
stUnivChar  ucdate[]        = { 'D', 'A', 'T', 'E', '\0' };
stUnivChar  ucpage[]        = { 'P', 'A', 'G', 'E', '\0' };
stUnivChar  ucfootnote[]    = { 'F', 'O', 'O', 'T', 'N', 'O', 'T', 'E', '\0' };
stUnivChar  uccolumn[]      = { 'C', 'O', 'L', 'U', 'M', 'N', ' ', 'X', '\0' };

stUnivString ustring[] =
{
    { ucnull, 0 },
    { ucdate, 4 },
    { ucpage, 4 },
    { ucfootnote, 8 },
    { uccolumn, 8 },
    { ucnull, 0 }
};


class clFieldImp : public clField
{
public:
    clFieldImp(uint8 id) :
        id_(id)
        {
        }

    virtual uint8   id() const;
    virtual void    release();
    virtual void    content(stUnivString&, APPColumn, TypeSpec&);
private:
    uint8   id_;
};


clField &clField::createField(scStream *str, uint8 id)
{
    clField *field = new clFieldImp(id);
    return *field;
}


uint8 clFieldImp::id() const
{
    return id_;
}


void clFieldImp::release()
{
    delete this;
}


void clFieldImp::content(stUnivString& ustr, APPColumn col, TypeSpec& )
{
    if (id_ < 5)
    {
        ustr = ustring[id_];
        if (id_ == 4)
        {
            stUnivChar *ptr = (stUnivChar *)ustr.ptr;
//          ptr[7]  = col->num() + '0';
        }
    }
    else
        ustr = ustring[0];
}
```

```
#ifndef __AGSYM_H_
#define __AGSYM_H_

#include "AGDoc.h"
#include "AGDC.h"
#include "AGText.h"

enum SYMTYPE
{
    ST_ANY = -1,
    ST_IMAGE,
    ST_TEXT
};


class CAGSym
{
public:
    CAGSym (SYMTYPE SymType);
    virtual ~CAGSym ();

    SYMTYPE GetSymType () const
                            { return (m_SymType); }
    virtual void Draw (CAGDC &dc) = 0;
    const CAGMatrix &GetMatrix () const
                            { return (m_Matrix); }
    virtual bool HitTest (POINT Pt) const = 0;
    virtual bool Read (CAGDocIO *pInput);
    void SetMatrix (const CAGMatrix &Matrix)
                            { m_Matrix = Matrix; }
    void Transform (const CAGMatrix &Matrix)
                            { m_Matrix *= Matrix; }
    virtual bool Write (CAGDocIO *pOutput);

protected:
    CAGMatrix   m_Matrix;

private:
    SYMTYPE     m_SymType;
};


class CAGSymImage : public CAGSym
{
public:
    CAGSymImage ();
    ~CAGSymImage ();

    void Draw (CAGDC &dc);
    void Free ();
    const RECT &GetDestRect () const
                            { return (m_DestRect); }
    bool HitTest (POINT Pt) const;
    void LoadDIB (const BITMAPINFOHEADER *pHdr, const BYTE *pBits);
    bool Read (CAGDocIO *pInput);
    void SetDestRect (const RECT &DestRect)
                            { m_DestRect = DestRect; }
    bool Write (CAGDocIO *pOutput);

protected:
    RECT                m_DestRect;
    BITMAPINFOHEADER    *m_pDIB;

};


class CAGSymText : public CAGSym, public CAGText
{
public:
    CAGSymText ();
    ~CAGSymText ();

    void Draw (CAGDC &dc);
    bool HitTest (POINT Pt) const;
    bool Read (CAGDocIO *pInput);
    bool Write (CAGDocIO *pOutput);
};

#endif //__AGSYM_H_
```

```
//===========================================================================//
//===========================================================================//
#include "stdafx.h"
#include "AGSym.h"
#include "AGDib.h"

#ifdef _AFX
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
#endif


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
CAGSym::CAGSym(SYMTYPE SymType)
{
    m_SymType = SymType;
}


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
CAGSym::~CAGSym()
{
}


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
bool CAGSym::Read(CAGDocIO *pInput)
{
    pInput->Read (&m_Matrix, sizeof (m_Matrix));
    return true;
}


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
bool CAGSym::Write(CAGDocIO *pOutput)
{
    pOutput->Write (&m_Matrix, sizeof (m_Matrix));
    return true;
}


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
CAGSymImage::CAGSymImage()
  : CAGSym (ST_IMAGE)
{
    m_pDIB = NULL;
    m_DestRect.left = m_DestRect.right = m_DestRect.top = m_DestRect.bottom = 0;
}


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
CAGSymImage::~CAGSymImage()
{
    Free();
}


//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
void CAGSymImage::Draw(CAGDC &dc)
{
    if (m_pDIB)
    {
        dc.PushModelingMatrix (m_Matrix);
        dc.StretchBlt (m_DestRect, DibPtr (m_pDIB), (BITMAPINFO *) m_pDIB);
        dc.PopModelingMatrix();
    }
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CAGSymImage::Free()
{
    if (m_pDIB)
    {
        free (m_pDIB);
        m_pDIB = NULL;
    }
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
bool CAGSymImage::HitTest(POINT Pt) const
{
    RECT r = m_DestRect;
    m_Matrix.Inverse ().Transform (&Pt, 1);
    return (::PtInRect (&r, Pt) != 0);
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CAGSymImage::LoadDIB(const BITMAPINFOHEADER *pHdr, const BYTE *pBits)
{
    Free();

    if (pHdr->biCompression == BI_RGB &&
        (pHdr->biBitCount == 1 || pHdr->biBitCount == 4 ||
        pHdr->biBitCount == 8 || pHdr->biBitCount == 24))
    {
        BITMAPINFOHEADER bi;
        bi = *pHdr;
        bi.biCompression = BI_RGB;

        if (bi.biSizeImage == 0)
            bi.biSizeImage = DibSizeImage (&bi);
        if (bi.biClrUsed == 0)
            bi.biClrUsed = DibNumColors (&bi);

        if ((m_pDIB = (BITMAPINFOHEADER *) malloc (DibSize (&bi))) != NULL)
        {
            *m_pDIB = bi;

            if (bi.biClrUsed)
            {
                memcpy ((void *) DibColors (m_pDIB), (void *) DibColors (pHdr),
                  DibPaletteSize (pHdr));
            }

            BYTE *pNewBits = (BYTE *) DibPtr (m_pDIB);
            const BYTE *pSrcBits;
            if (pBits)
                pSrcBits = pBits;
            else
                pSrcBits = (BYTE *) (DibColors(pHdr) + bi.biClrUsed);

            memcpy (pNewBits, pSrcBits, bi.biSizeImage);
        }
    }
}

//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
bool CAGSymImage::Read(CAGDocIO *pInput)
```

```
{
    Free();

    bool bReturn = CAGSym::Read(pInput);

    DWORD dwSize = 0;
    pInput->Read(&dwSize, sizeof (dwSize));

    if (dwSize > 0)
    {
        m_pDIB = (BITMAPINFOHEADER *) malloc (dwSize);
        pInput->Read(m_pDIB, dwSize);
        pInput->Read(&m_DestRect, sizeof (m_DestRect));
    }

    return bReturn;
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
bool CAGSymImage::Write(CAGDocIO *pOutput)
{
    bool bReturn = CAGSym::Write(pOutput);

    if (m_pDIB)
    {
        DWORD dwSize = DibSize (m_pDIB);
        pOutput->Write (&dwSize, sizeof (dwSize));
        pOutput->Write (m_pDIB, dwSize);
        pOutput->Write (&m_DestRect, sizeof (m_DestRect));
    }
    else
    {
        DWORD dwSize = 0;
        pOutput->Write (&dwSize, sizeof (dwSize));
    }

    return bReturn;
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
CAGSymText::CAGSymText()
  : CAGSym (ST_TEXT)
{
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
CAGSymText::~CAGSymText()
{
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CAGSymText::Draw(CAGDC &dc)
{
    dc.PushModelingMatrix(m_Matrix);
    DrawColumn(dc);
    dc.PopModelingMatrix();
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
bool CAGSymText::HitTest(POINT Pt) const
{
    RECT r = m_DestRect;
    m_Matrix.Inverse().Transform (&Pt, 1);
    return (::PtInRect (&r, Pt) != 0);
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
bool CAGSymText::Read(CAGDocIO *pInput)
{
    bool bReturn = CAGSym::Read(pInput);
    ReadColumn(pInput);
    return (bReturn);
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
bool CAGSymText::Write(CAGDocIO *pOutput)
{
    bool bReturn = CAGSym::Write(pOutput);
    WriteColumn(pOutput);
    return (bReturn);
}
```

09624340.073400

```
#ifndef __AGPAGE_H_
#define __AGPAGE_H_

#include "AGDoc.h"
#include "AGSym.h"
#include "AGDC.h"

class CAGLayer;

#define MAX_AGLAYER 10

class CAGPage
{
public:
    CAGPage(int cx = 0, int cy = 0);
    ~CAGPage();

    void AppendLayer (CAGLayer *pLayer)             { m_pLayers[m_nLayers++] = pLayer; }
    void Draw (CAGDC &dc) const;
    CAGSym *FindSymbolByPoint (POINT Pt, SYMTYPE SymType = ST_ANY) const;
    void GetFonts (LOGFONTARRAY &lfArray);
    CAGLayer *GetLayer (int nLayer) const;
    int GetNumLayers() const                         { return (m_nLayers); }
    const char *GetPageName () const                 { return (m_szPageName); }
    void GetPageSize(SIZE *pSize) const              { *pSize = m_PageSize; }
    bool IsEmpty () const;
    bool Read(CAGDocIO *pInput);
    void SetPageName (const char *pszPageName)       { lstrcpyn (m_szPageName, pszPageName, sizeof (m
_szPageName)); }
    bool Write(CAGDocIO *pOutput) const;

protected:
    int         m_nLayers;
    SIZE        m_PageSize;
    CAGLayer    *m_pLayers[MAX_AGLAYER];
    char        m_szPageName[255];
};

#endif //__AGPAGE_H_
```

```
//==========================================================================//
//==========================================================================//
#include "stdafx.h"
#include "AGPage.h"
#include "AGLayer.h"

#ifdef _AFX
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
#endif


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
CAGPage::CAGPage(int cx, int cy)
{
    m_PageSize.cx = cx;
    m_PageSize.cy = cy;
    m_nLayers = 0;
    for (int i = 0; i < MAX_AGLAYER; i++)
        m_pLayers[i] = NULL;
    m_szPageName[0] = 0;
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
CAGPage::~CAGPage()
{
    for (int i = 0; i < m_nLayers; i++)
        delete m_pLayers[i];


void CAGPage::Draw (CAGDC &dc) const

    for (int i = 0; i < m_nLayers; i++)
        m_pLayers[i]->Draw (dc);
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
CAGSym *CAGPage::FindSymbolByPoint(POINT Pt, SYMTYPE SymType) const

    CAGSym *pSym = NULL;

    for (int i = GetNumLayers (); i > 0 && pSym == NULL; i--)
    {
        CAGLayer *pAGLayer = GetLayer (i);
        pSym = pAGLayer->FindSymbolByPoint (Pt, SymType);
    }
    return (pSym);
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CAGPage::GetFonts(LOGFONTARRAY &lfArray)
{
    for (int i = 0; i < m_nLayers; i++)
        m_pLayers[i]->GetFonts (lfArray);
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
CAGLayer *CAGPage::GetLayer(int nLayer) const
{
    if (nLayer == 0 || nLayer > m_nLayers)
```

```
        return NULL;
    else
        return m_pLayers[nLayer - 1];
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
bool CAGPage::IsEmpty() const
{
    bool bEmpty = true;

    for (int i = 0; i < m_nLayers; i++)
    {
        if (! m_pLayers[i]->IsEmpty())
        {
            bEmpty = false;
            break;
        }
    }

    return bEmpty;
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
bool CAGPage::Read(CAGDocIO *pInput)
{
    BYTE bPageLen;
    pInput->Read(&bPageLen, sizeof (bPageLen));
    if (bPageLen > 0)
    {
        pInput->Read(m_szPageName, bPageLen);
        m_szPageName[bPageLen] = 0;
    }
    else
        m_szPageName[0] = 0;

    pInput->Read(&m_PageSize, sizeof (m_PageSize));
    pInput->Read(&m_nLayers, sizeof (m_nLayers));
    for (int i = 0; i < m_nLayers; i++)
    {
        CAGLayer *pLayer = new CAGLayer();
        pLayer->Read(pInput);
        m_pLayers[i] = pLayer;
    }

    return true;
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
bool CAGPage::Write(CAGDocIO *pOutput) const
{
    bool bReturn = true;

    BYTE bPageLen = (BYTE)lstrlen(m_szPageName);
    pOutput->Write(&bPageLen, sizeof(bPageLen));
    if (bPageLen > 0)
        pOutput->Write(m_szPageName, bPageLen);
    pOutput->Write(&m_PageSize, sizeof (m_PageSize));
    pOutput->Write(&m_nLayers, sizeof (m_nLayers));

    for (int i = 0; i < m_nLayers; i++)
    {
        CAGLayer *pLayer = m_pLayers[i];
        pLayer->Write(pOutput);
    }

    return bReturn;
}
```

```
#ifndef __AGMATRIX_H_
#define __AGMATRIX_H_

class CAGMatrix
{
public:
    CAGMatrix ()
      { Unity (); }
    CAGMatrix (double m11, double m12, double m21, double m22,
      double m31 = 0, double m32 = 0)
      { m_11 = m11; m_12 = m12; m_21 = m21; m_22 = m22; m_31 = m31; m_32 = m32; }
    CAGMatrix operator * (const CAGMatrix &Matrix) const;
    void operator *= (const CAGMatrix &Matrix)
      { *this = *this * Matrix; }
    bool GetRotation (int &Angle) const;
    CAGMatrix Inverse () const;
    bool IsUnity () const
      { return (m_11 == 1 && m_12 == 0 && m_21 == 0 && m_22 == 1 &&
      m_31 == 0 && m_32 == 0); }
    void Rotate (double Angle, double x = 0, double y = 0);
    void Scale (double xScale, double yScale, double x = 0, double y = 0);
    void ScaleAndCenter (const RECT &DestRect, const RECT &SrcRect,
      bool bFlip = false);
    void SetMatrix (double m11, double m12, double m21, double m22,
      double m31 = 0, double m32 = 0)
      { m_11 = m11; m_12 = m12; m_21 = m21; m_22 = m22; m_31 = m31; m_32 = m32; }
    void Transform (POINT *pPts, int nCount, bool bTranslate = true) const;
    void Transform (RECT *pRect, bool bTranslate = true) const
      { Transform ((POINT *) pRect, 2, bTranslate); }
    void Translate (double x, double y)
      { m_31 += x; m_32 += y; }
    void Translate (int x, int y)
      { m_31 += (double) x; m_32 += (double) y; }
    void Unity ()
      { m_11 = m_22 = 1; m_12 = m_21 = m_31 = m_32 = 0; }

public:
    double  m_11;
    double  m_12;
    double  m_21;
    double  m_22;
    double  m_31;
    double  m_32;
};

#endif //__AGMATRIX_H_
```

```
//==========================================================================//
//==========================================================================//
#include "stdafx.h"
#include "AGMatrix.h"

#include <math.h>

#ifdef _AFX
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
#endif

#define PI  3.1415926535897932
#define ABS(Value) ((Value) > 0 ? (Value) : (-(Value)))

int Arctan (int X, int Y, int xScale, int yScale)
{
    int Angle;
    int Quad;
    if (X > 0)
        Quad = (Y > 0) ? 4 : 1;
    else if (X < 0)
        Quad = (Y < 0) ? 2 : 3;
    else
        Quad = ((Y == 0) ? 5 : ((Y < 0) ? 2 : 3));

    if (Quad == 5)
        Angle = 0;
    else
    {
        Y = ABS (Y);
        if (Quad > 2)
            X = -X;
        if (yScale == 0)
            Angle = 0;
        else if (xScale == 0)
            Angle = 900;
        else
        {
            double datan = atan2 ((double)Y * (double)xScale,
              (double)X * (double)yScale);
            datan *= (57.29578 * 10);
            Angle = (int)(datan + ((datan >= 0) ? 0.5 : -0.5));
        }
        if (Quad > 2)
            Angle += 1800;
    }
    return (Angle);
}

//--------------------------------------------------------------------------//
// Double to int with rounding.                                             //
//--------------------------------------------------------------------------//
inline int dtoi (double x)
{
    return ((int) (x > 0 ? x + 0.5 : x - 0.5));
}

//--------------------------------------------------------------------------//
// Multiply the current matrix with the specified matrix the return a new   //
// matrix.                                                                  //
//--------------------------------------------------------------------------//
CAGMatrix CAGMatrix::operator * (const CAGMatrix &Matrix) const
{
    return (CAGMatrix (m_11 * Matrix.m_11 + m_12 * Matrix.m_21,
                       m_11 * Matrix.m_12 + m_12 * Matrix.m_22,
                       m_21 * Matrix.m_11 + m_22 * Matrix.m_21,
                       m_21 * Matrix.m_12 + m_22 * Matrix.m_22,
                       m_31 * Matrix.m_11 + m_32 * Matrix.m_21 + Matrix.m_31,
                       m_31 * Matrix.m_12 + m_32 * Matrix.m_22 + Matrix.m_32));
}

//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
bool CAGMatrix::GetRotation (int &Angle) const
{
    bool bRotated = false;
    if ((m_12 > 0.0000005) || (-m_12 > 0.0000005) ||
      (m_21 > 0.0000005) || (-m_21 > 0.0000005) ||
      (m_11 < 0.0) || (m_22 < 0.0))
    {
        POINT Pts[3];
        Pts[0].x = Pts[0].y = Pts[1].y = Pts[2].x = 0;
        Pts[1].x = Pts[2].y = 100000;
        Transform (Pts, 3);

        Angle = Arctan (Pts[1].x - Pts[0].x,Pts[1].y - Pts[0].y,1,1);
        int Angle2 = Arctan (Pts[2].x - Pts[0].x,Pts[2].y - Pts[0].y,1,1);
        int AngleRes = Angle2 - Angle;
        if (AngleRes < 0)
            AngleRes += 3600;
        bRotated = ((AngleRes >= 2698) && (AngleRes <= 2702));
    }
    return (bRotated);
}

//--------------------------------------------------------------------------//
// Return the inverse of the matrix.                                        //
//                          1                                               //
// The inverse of a b is   ---  *   d -b                                    //
//                c d      det     -c  a                                    //
//                                                                          //
// The determinant is calculated as a*d - b*c                               //
//--------------------------------------------------------------------------//
CAGMatrix CAGMatrix::Inverse () const
{
    CAGMatrix Inverse;
    double det = m_11 * m_22 - m_12 * m_21;
    if (det == 0.0)
        Inverse = *this;
    else
    {
        det = 1.0 / det;
        Inverse.m_11 = det * m_22;
        Inverse.m_12 = det * -m_12;
        Inverse.m_21 = det * -m_21;
        Inverse.m_22 = det * m_11;
    }

    Inverse.m_31 = -m_31 * Inverse.m_11 + -m_32 * Inverse.m_21;
    Inverse.m_32 = -m_31 * Inverse.m_12 + -m_32 * Inverse.m_22;

    return (Inverse);
}

//--------------------------------------------------------------------------//
// Rotate the matrix.                                                       //
//--------------------------------------------------------------------------//
void CAGMatrix::Rotate (double Angle, double x, double y)
{
    double AngleRadian = Angle * PI / 180;

    Translate (-x, -y);
    double CosAngle = cos (AngleRadian);
    double SinAngle = sin (AngleRadian);
    CAGMatrix Temp (CosAngle, -SinAngle, SinAngle, CosAngle);
    *this *= Temp;
    Translate (x, y);
}

//--------------------------------------------------------------------------//
// Scale the matrix.                                                        //
//--------------------------------------------------------------------------//
```

```
void CAGMatrix::Scale (double xScale, double yScale, double x, double y)
{
    Translate (-x, -y);
    m_11 *= xScale;
    m_12 *= yScale;
    m_21 *= xScale;
    m_22 *= yScale;
    m_31 *= xScale;
    m_32 *= yScale;
    Translate (x, y);
}

//--------------------------------------------------------------------------//
// Calculate the matrix to scale and center the souce rectangle within the  //
// destination rectangle.                                                   //
//--------------------------------------------------------------------------//
void CAGMatrix::ScaleAndCenter (const RECT &DestRect, const RECT &SrcRect,
  bool bFlip)
{
    double xScale = (double) WIDTH (DestRect) / (double) WIDTH (SrcRect);
    double yScale = (double) HEIGHT (DestRect) / (double) HEIGHT (SrcRect);
    if (xScale < yScale)
        yScale = xScale;
    else
        xScale = yScale;

    SetMatrix (xScale, 0, 0, yScale);
    RECT New = SrcRect;
    Transform (&New);
    Translate (DestRect.left - New.left + ((WIDTH (DestRect) - WIDTH (New)) / 2),
      DestRect.top - New.top + ((HEIGHT (DestRect) - HEIGHT (New)) / 2));

    if (bFlip)
    {
        Scale (-1, -1, (DestRect.left + DestRect.right) / 2,
          (DestRect.top + DestRect.bottom) / 2);
    }
}

//--------------------------------------------------------------------------//
// Transform the specified points.                                          //
//--------------------------------------------------------------------------//
void CAGMatrix::Transform (POINT *pPts, int nCount, bool bTranslate) const
{
    if (bTranslate)
    {
        while (nCount--)
        {
            int x = dtoi ((double) pPts->x * m_11 + (double) pPts->y * m_21 + m_31);
            pPts->y = dtoi ((double) pPts->x * m_12 + (double) pPts->y * m_22 + m_32);
            pPts->x = x;
            pPts++;
        }
    }
    else
    {
        while (nCount--)
        {
            int x = dtoi ((double) pPts->x * m_11 + (double) pPts->y * m_21);
            pPts->y = dtoi ((double) pPts->x * m_12 + (double) pPts->y * m_22);
            pPts->x = x;
            pPts++;
        }
    }
}
```

```
#ifndef __AGLAYER_H_
#define __AGLAYER_H_

#include "AGDoc.h"
#include "AGSym.h"
#include "AGDC.h"

class CAGSym;

#define MAX_AGSYMBOLS 100



class CAGLayer
{
public:
    CAGLayer();
    ~CAGLayer();

    void AppendSymbol (CAGSym *pSym);
    void Draw (CAGDC &dc);
    CAGSym *FindFirstSymbolByType (SYMTYPE SymType) const;
    CAGSym *FindSymbolByPoint (POINT Pt, SYMTYPE SymType = ST_ANY) const;
    void GetFonts (LOGFONTARRAY &lfArray);
    bool IsEmpty ();
    bool Read (CAGDocIO *pInput);
    bool Write (CAGDocIO *pOutput);

protected:

    CAGSym  *m_pSymbols[MAX_AGSYMBOLS];


#endif //__AGLAYER_H_
```

```
//=========================================================================//
//=========================================================================//
#include "stdafx.h"
#include "AGLayer.h"
#include "AGSym.h"

#ifdef _AFX
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
#endif


//-------------------------------------------------------------------------//
//-------------------------------------------------------------------------//
CAGLayer::CAGLayer()
{
    for (int i = 0; i < MAX_AGSYMBOLS; i++)
        m_pSymbols[i] = NULL;
}


//-------------------------------------------------------------------------//
//-------------------------------------------------------------------------//
CAGLayer::~CAGLayer()
{
    for (int i = 0; m_pSymbols[i]; i++)
        delete m_pSymbols[i];
}


//-------------------------------------------------------------------------//
//-------------------------------------------------------------------------//
void CAGLayer::AppendSymbol(CAGSym *pSym)
{
    for (int i = 0; m_pSymbols[i]; i++)
        ;

    m_pSymbols[i] = pSym;
}


//-------------------------------------------------------------------------//
//-------------------------------------------------------------------------//
void CAGLayer::Draw(CAGDC &dc)
{
    for (int i = 0; m_pSymbols[i]; i++)
        m_pSymbols[i]->Draw (dc);
}


//-------------------------------------------------------------------------//
//-------------------------------------------------------------------------//
CAGSym *CAGLayer::FindFirstSymbolByType(SYMTYPE SymType) const
{
    CAGSym *pSym = NULL;

    for (int i = 0; m_pSymbols[i] && pSym == NULL; i++)
    {
        if (m_pSymbols[i]->GetSymType() == SymType)
            pSym = m_pSymbols[i];
    }
    return (pSym);
}


//-------------------------------------------------------------------------//
//-------------------------------------------------------------------------//
CAGSym *CAGLayer::FindSymbolByPoint(POINT Pt, SYMTYPE SymType) const
{
    CAGSym *pSym = NULL;
    for (int i = 0; m_pSymbols[i] && pSym == NULL; i++)
    {
        if (SymType == ST_ANY || m_pSymbols[i]->GetSymType() == SymType)
        {
            if (m_pSymbols[i]->HitTest(Pt))
                pSym = m_pSymbols[i];
        }
    }
    return (pSym);
}

//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CAGLayer::GetFonts(LOGFONTARRAY &lfArray)
{
    for (int i = 0; m_pSymbols[i]; i++)
    {
        if (m_pSymbols[i]->GetSymType() == ST_TEXT)
        {
            CAGSymText *pText = (CAGSymText *) m_pSymbols[i];
            pText->GetFonts(lfArray);
        }
    }
}

//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
bool CAGLayer::IsEmpty()
{
    if (m_pSymbols[0])
        return false;
    else
        return true;
}

//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
bool CAGLayer::Read(CAGDocIO *pInput)
{
    int nSymbols;
    pInput->Read(&nSymbols, sizeof(nSymbols));
    for (int i = 0; i < nSymbols; i++)
    {
        SYMTYPE SymType;
        pInput->Read(&SymType, sizeof(SymType));

        CAGSym *pSym = NULL;
        switch (SymType)
        {
            case ST_IMAGE:
                pSym = new CAGSymImage();
                break;

            case ST_TEXT:
                pSym = new CAGSymText();
                break;
        }

        if (pSym)
        {
            pSym->Read(pInput);
            m_pSymbols[i] = pSym;
        }
    }

    return true;
}
```

```
//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
bool CAGLayer::Write(CAGDocIO *pOutput)
{
    bool bReturn = true;

    int nSymbols = 0;
    for (nSymbols; m_pSymbols[nSymbols]; nSymbols++)
        ;

    pOutput->Write(&nSymbols, sizeof (nSymbols));

    for (int i = 0; i < nSymbols; i++)
    {
        CAGSym *pSym = m_pSymbols[i];
        SYMTYPE SymType = pSym->GetSymType();
        pOutput->Write(&SymType, sizeof (SymType));
        pSym->Write(pOutput);
    }

    return bReturn;
}
```

```
#ifndef __AGDOC_H_
#define __AGDOC_H_

#include <iostream.h>
#include "zutil.h"

#include "AGDC.h"
class CAGPage;
typedef std::vector<LOGFONT> LOGFONTARRAY;

#define MAX_AGPAGE  4
#define Z_BUFSIZE   16384


enum AGDOCTYPE
{
    DOC_DEFAULT,
    DOC_CARDHV,
    DOC_CARDHH,
    DOC_CARDFV,
    DOC_CARDFH,
};

enum PRINTSIDE
{
    PRINT_OUTSIDE,
    PRINT_INSIDE,
    PRINT_BOTH
};

class CAGDocIO
{
public:
    CAGDocIO (istream *pInput);
    CAGDocIO (ostream *pOutput);

    void Close ();
    void Read (void *pData, DWORD dwSize);
    void Write (const void *pData, DWORD dwSize);

protected:
    bool        m_bInput;
    bool        m_bEOF;
    istream     *m_pInput;
    ostream     *m_pOutput;
    z_stream    m_zstream;
    BYTE        m_zBuf[Z_BUFSIZE];
    int         m_zErr;
};



class CAGDoc
{
public:
    CAGDoc (AGDOCTYPE AGDocType = DOC_DEFAULT);
    ~CAGDoc ();

    void AppendPage (CAGPage *pPage)
        { m_pPages[m_nPages++] = pPage; m_nCurPage = m_nPages; };
    void Free ();
    AGDOCTYPE GetDocType ()
        { return (m_AGDocType); }
    int GetNumPages ()
        { return (m_nPages); }
    CAGPage *GetCurrentPage ()
        { return (GetPage(m_nCurPage)); }
    int GetCurrentPageNum ()
        { return (m_nCurPage); }
    void GetFonts (LOGFONTARRAY &lfArray);
    CAGPage *GetPage (int nPage);
    bool PrintCardQuarter (char *pszDriver, char *pszDevice, char *pszOutput,
      DEVMODE *pDevMode, const char *pszFileName = NULL);
    bool PrintCardSingle (PRINTSIDE PrintSide, char *pszDriver,
      char *pszDevice, char *pszOutput, DEVMODE *pDevMode, bool &bRotated,
      const char *pszFileName = NULL);
    bool Read(istream &input);
    void SetCurrentPage (int nPage)
        { if (nPage > 0 && nPage <= m_nPages) m_nCurPage = nPage; }
    bool Write (ostream &output);

protected:
    void GetQFPageRect (int nPage, bool bPortrait, bool bMaxMargin,
      CAGDC *pDC, CAGDCInfo &di, RECT &DestRect, bool &bFlip, bool &bRotated);
    void GetSFPageRect (int nPage, bool bPortrait, bool bMaxMargin,
      CAGDC *pDC, CAGDCInfo &di, RECT &DestRect, bool &bFlip, bool &bRotated);

protected:
    AGDOCTYPE   m_AGDocType;
    int         m_nPages;
    int         m_nCurPage;
    CAGPage     *m_pPages[MAX_AGPAGE];
};

#endif //__AGDOC_H_
```

```
//==========================================================================//
//==========================================================================//
#include "stdafx.h"
#include "AGDoc.h"
#include "AGPage.h"
#include "AGDC.h"

#ifdef _AFX
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
#endif

#define AGDOC_ID        "Ag"
#define AGDOC_VERSION   1

typedef struct
{
    BYTE abId[2];
    BYTE bVer;
} AGHDR;


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
CAGDoc::CAGDoc(AGDOCTYPE AGDocType)
{
    m_AGDocType = AGDocType;
    m_nCurPage = 0;
    m_nPages = 0;
    for (int i = 0; i < MAX_AGPAGE; i++)
        m_pPages[i] = NULL;
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
CAGDoc::~CAGDoc()
{
    Free();
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CAGDoc::Free()
{
    for (int i = 0; i < m_nPages; i++)
    {
        delete m_pPages[i];
        m_pPages[i] = NULL;
    }
    m_nCurPage = 0;
    m_nPages = 0;
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CAGDoc::GetFonts(LOGFONTARRAY &lfArray)
{
    for (int i = 0; i < m_nPages; i++)
        m_pPages[i]->GetFonts(lfArray);
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
CAGPage *CAGDoc::GetPage(int nPage)
{
    if (nPage == 0 || nPage > m_nPages)
```

```
        return (NULL);
    else
        return (m_pPages[nPage - 1]);
}


//-------------------------------------------------------------------------//
//-------------------------------------------------------------------------//
void CAGDoc::GetQFPageRect(int nPage, bool bPortrait, bool bMaxMargin, CAGDC *pDC,
                            CAGDCInfo &di, RECT &DestRect, bool &bFlip, bool &bRotated)
{
    //---------------------------------------------------------------------//
    // Set a small minimum margin because HP DeskJets report a .04 top      //
    // margin but obviously can't print that close to the edge of the page. //
    //---------------------------------------------------------------------//
    int nMinMarginX = di.m_nLogPixelsX / 10;
    int nMinMarginY = di.m_nLogPixelsY / 10;

    RECT Margin =
    {
        max(di.m_PrintOffset.cx, nMinMarginX),
        max(di.m_PrintOffset.cy, nMinMarginY),
        max((di.m_PhysPageSize.cx - di.m_nHorzRes - di.m_PrintOffset.cx), nMinMarginX),
        max((di.m_PhysPageSize.cy - di.m_nVertRes - di.m_PrintOffset.cy), nMinMarginY)
    };

    SIZE MarginMax =
    {
        max(Margin.left, Margin.right),
        max(Margin.top, Margin.bottom)
    };

    bRotated = false;
    if ((bPortrait && (Margin.top < Margin.bottom)) ||
        (! bPortrait && (Margin.left < Margin.right)))
    {
        nPage = ((nPage + 1) % 4) + 1;
        bRotated = true;
    }

    bFlip = false;

    switch (nPage)
    {
        case 1:
        {
            if (bMaxMargin)
            {
                DestRect.left = di.m_PhysPageSize.cx / 2 + MarginMax.cx;
                DestRect.right = di.m_PhysPageSize.cx - MarginMax.cx;
            }
            else
            {
                DestRect.left = di.m_PhysPageSize.cx / 2 + Margin.right;
                DestRect.right = di.m_PhysPageSize.cx - Margin.right;
            }
            DestRect.top = di.m_PhysPageSize.cy / 2 + Margin.bottom;
            DestRect.bottom = di.m_PhysPageSize.cy - Margin.bottom;
            break;
        }

        case 2:
        {
            if (bPortrait)
            {
                if (bMaxMargin)
                {
                    DestRect.left = di.m_PhysPageSize.cx / 2 + MarginMax.cx;
                    DestRect.right = di.m_PhysPageSize.cx - MarginMax.cx;
                }
                else
                {
                    DestRect.left = di.m_PhysPageSize.cx / 2 + Margin.right;
                    DestRect.right = di.m_PhysPageSize.cx - Margin.right;
                }
                DestRect.top = Margin.top;
                DestRect.bottom = di.m_PhysPageSize.cy / 2 - Margin.top;
                bFlip = true;
            }
            else
            {
                DestRect.left = Margin.left;
                DestRect.right = di.m_PhysPageSize.cx / 2 - Margin.left;
                if (bMaxMargin)
                {
                    DestRect.top = di.m_PhysPageSize.cy / 2 + MarginMax.cy;
                    DestRect.bottom = di.m_PhysPageSize.cy - MarginMax.cy;
                }
                else
                {
                    DestRect.top = di.m_PhysPageSize.cy / 2 + Margin.bottom;
                    DestRect.bottom = di.m_PhysPageSize.cy - Margin.bottom;
                }
                if (!bRotated)
                    bFlip = true;
            }
            break;
        }

        case 3:
        {
            if (bPortrait)
            {
                if (bMaxMargin)
                {
                    DestRect.left = MarginMax.cx;
                    DestRect.right = di.m_PhysPageSize.cx / 2 - MarginMax.cx;
                }
                else
                {
                    DestRect.left = Margin.left;
                    DestRect.right = di.m_PhysPageSize.cx / 2 - Margin.left;
                }
                DestRect.top = Margin.top;
                DestRect.bottom = di.m_PhysPageSize.cy / 2 - Margin.top;
            }
            else
            {
                DestRect.left = Margin.left;
                DestRect.right = di.m_PhysPageSize.cx / 2 - Margin.left;
                if (bMaxMargin)
                {
                    DestRect.top = MarginMax.cy;
                    DestRect.bottom = di.m_PhysPageSize.cy / 2 - MarginMax.cy;
                }
                else
                {
                    DestRect.top = Margin.top;
                    DestRect.bottom = di.m_PhysPageSize.cy / 2 - Margin.top;
                }
            }
            bFlip = true;
            break;
        }

        case 4:
        {
            if (bPortrait)
            {
                if (bMaxMargin)
                {
                    DestRect.left = MarginMax.cx;
                    DestRect.right = di.m_PhysPageSize.cx / 2 - MarginMax.cx;
                }
                else
                {
                    DestRect.left = Margin.left;
                    DestRect.right = di.m_PhysPageSize.cx / 2 - Margin.left;
                }
                DestRect.top = di.m_PhysPageSize.cy / 2 + Margin.bottom;
                DestRect.bottom = di.m_PhysPageSize.cy - Margin.bottom;
            }
            else
            {
                if (bMaxMargin)
                {
                    DestRect.left = di.m_PhysPageSize.cx / 2 + MarginMax.cx;
                    DestRect.right = di.m_PhysPageSize.cx - MarginMax.cx;
                }
                else
                {
                    DestRect.left = di.m_PhysPageSize.cx / 2 + Margin.right;
                    DestRect.right = di.m_PhysPageSize.cx - Margin.right;
                }
                DestRect.top = Margin.top;
                DestRect.bottom = di.m_PhysPageSize.cy / 2 - Margin.top;
                if (!bRotated)
                    bFlip = true;
            }
            break;
        }
    }

    pDC->DPAtoVPA(&DestRect);
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CAGDoc::GetSFPageRect(int nPage, bool bPortrait, bool bMaxMargin, CAGDC *pDC,
                           CAGDCInfo &di, RECT &DestRect, bool &bFlip, bool &bRotated)
{
    //----------------------------------------------------------------------//
    // Set a small minimum margin because HP DeskJets report a .04 top      //
    // margin but obviously can't print that close to the edge of the page. //
    //----------------------------------------------------------------------//
    int nMinMarginX = di.m_nLogPixelsX / 10;
    int nMinMarginY = di.m_nLogPixelsY / 10;

    RECT Margin =
    {
            max(di.m_PrintOffset.cx, nMinMarginX),
            max(di.m_PrintOffset.cy, nMinMarginY),
            max((di.m_PhysPageSize.cx - di.m_nHorzRes - di.m_PrintOffset.cx), nMinMarginX),
            max((di.m_PhysPageSize.cy - di.m_nVertRes - di.m_PrintOffset.cy), nMinMarginY)
    };

    SIZE MarginMax =
    {
        max(Margin.left, Margin.right),
        max(Margin.top, Margin.bottom)
    };

    bRotated = false;
    if ((bPortrait && (Margin.top < Margin.bottom)) ||
        (!bPortrait && (Margin.left < Margin.right)))
    {
        nPage = 4 - nPage + 1;
        bRotated = true;
    }

    bFlip = false;

    switch (nPage)
    {
        case 1:
        case 3:
        {
            if (bPortrait)
            {
                DestRect.left = MarginMax.cx;
                DestRect.right = di.m_PhysPageSize.cx - MarginMax.cx;
                if (bMaxMargin)
                {
                    DestRect.top = di.m_PhysPageSize.cy / 2 + MarginMax.cy;
                    DestRect.bottom = di.m_PhysPageSize.cy - MarginMax.cy;
                }
                else
                {
                    DestRect.top = di.m_PhysPageSize.cy / 2 + Margin.bottom;
                    DestRect.bottom = di.m_PhysPageSize.cy - Margin.bottom;
                }
                if (bRotated && nPage == 3)
                    bFlip = true;
            }
            else
            {
                if (bMaxMargin)
                {
                    DestRect.left = di.m_PhysPageSize.cx / 2 + MarginMax.cx;
                    DestRect.right = di.m_PhysPageSize.cx - MarginMax.cx;
                }
                else
                {
                    DestRect.left = di.m_PhysPageSize.cx / 2 + Margin.right;
                    DestRect.right = di.m_PhysPageSize.cx - Margin.right;
                }
                DestRect.top = MarginMax.cy;
                DestRect.bottom = di.m_PhysPageSize.cy - MarginMax.cy;
                bFlip = bRotated;
            }
            break;
        }

        case 2:
        case 4:
        {
            if (bPortrait)
            {
                DestRect.left = MarginMax.cx;
                DestRect.right = di.m_PhysPageSize.cx - MarginMax.cx;
                if (bMaxMargin)
                {
                    DestRect.top = MarginMax.cy;
                    DestRect.bottom = di.m_PhysPageSize.cy / 2 - MarginMax.cy;
                }
                else
                {
                    DestRect.top = Margin.top;
                    DestRect.bottom = di.m_PhysPageSize.cy / 2 - Margin.top;
                }
                if (nPage == 4 || bRotated)
                    bFlip = true;
            }
            else
            {
                if (bMaxMargin)
                {
                    DestRect.left = MarginMax.cx;
                    DestRect.right = di.m_PhysPageSize.cx / 2 - MarginMax.cx;
                }
                else
                {
                    DestRect.left = Margin.left;
                    DestRect.right = di.m_PhysPageSize.cx / 2 - Margin.left;
                }
                DestRect.top = MarginMax.cy;
                DestRect.bottom = di.m_PhysPageSize.cy - MarginMax.cy;

                bFlip = bRotated;
            }
            break;
        }
    }

    pDC->DPAtoVPA(&DestRect);
}


//--------------------------------------------------------------------------//
// Print quarter-fold card.                                                 //
//--------------------------------------------------------------------------//
bool CAGDoc::PrintCardQuarter(char *pszDriver, char *pszDevice, char *pszOutput,
                               DEVMODE *pDevMode, const char *pszFileName)
{
    if (m_nPages != 4)
        return (false);

    bool bPortrait = (m_AGDocType == DOC_CARDFV || m_AGDocType == DOC_CARDHV);

    if (pDevMode)
    {
        if (bPortrait)
            pDevMode->dmOrientation = DMORIENT_PORTRAIT;
        else
            pDevMode->dmOrientation = DMORIENT_LANDSCAPE;
        pDevMode->dmFields |= DM_ORIENTATION;
    }

    CAGDC *pDC = new CAGDC(pszDriver, pszDevice, pszOutput, pDevMode);
    CAGDCInfo di = pDC->GetDeviceInfo();

    if (pDevMode && pDevMode->dmOrientation == DMORIENT_PORTRAIT &&
        di.m_nHorzSize > di.m_nVertSize)
    {
        delete pDC;
        pDevMode->dmOrientation = DMORIENT_LANDSCAPE;
        pDC = new CAGDC(pszDriver, pszDevice, pszOutput, pDevMode);
        di = pDC->GetDeviceInfo();
    }
    else if (pDevMode && pDevMode->dmOrientation == DMORIENT_LANDSCAPE &&
        di.m_nVertSize > di.m_nHorzSize)
    {
        delete pDC;
        pDevMode->dmOrientation = DMORIENT_PORTRAIT;
        pDC = new CAGDC(pszDriver, pszDevice, pszOutput, pDevMode);
        di = pDC->GetDeviceInfo();
    }

    RECT        DestRect;
    RECT        SrcRect;
    SIZE        sizePage;
    CAGMatrix   ViewMatrix;
    bool        bFlip;
    bool        bRotated;

    if (!pDC->StartDoc("Create and Print"))
    {
        delete pDC;
        return (false);
    }

    if (!pDC->StartPage())
    {
        pDC->AbortDoc();
        delete pDC;
        return (false);
    }

    //--------------------------------------------------------------------------//
    // Front                                                                    //
    //--------------------------------------------------------------------------//
    CAGPage *pPage = GetPage(1);
    pPage->GetPageSize(&sizePage);
    ::SetRect(&SrcRect, 0, 0, sizePage.cx, sizePage.cy);
    GetQFPageRect(1, bPortrait, false, pDC, di, DestRect, bFlip, bRotated);
    ViewMatrix.ScaleAndCenter(DestRect, SrcRect, bFlip);
    pDC->SetViewingMatrix(ViewMatrix);
    pPage->Draw(*pDC);

    bool bPage2Empty = GetPage(2)->IsEmpty();
    bool bPage3Empty = GetPage(3)->IsEmpty();

    //----------------------------------------------------------------------//
    // Inside Left / Top                                                    //
    //----------------------------------------------------------------------//
    pPage = GetPage(2);
    pPage->GetPageSize(&sizePage);
    ::SetRect(&SrcRect, 0, 0, sizePage.cx, sizePage.cy);
    GetQFPageRect(2, bPortrait, ! bPage3Empty, pDC, di, DestRect, bFlip,
                    bRotated);
    ViewMatrix.ScaleAndCenter(DestRect, SrcRect, bFlip);
    pDC->SetViewingMatrix(ViewMatrix);
    pPage->Draw(*pDC);

    //----------------------------------------------------------------------//
    // Inside Right / Bottom                                                //
    //----------------------------------------------------------------------//
    pPage = GetPage(3);
    pPage->GetPageSize(&sizePage);
    ::SetRect(&SrcRect, 0, 0, sizePage.cx, sizePage.cy);
    GetQFPageRect(3, bPortrait, ! bPage2Empty, pDC, di, DestRect, bFlip,
                    bRotated);
    ViewMatrix.ScaleAndCenter(DestRect, SrcRect, bFlip);
    pDC->SetViewingMatrix(ViewMatrix);
    pPage->Draw(*pDC);

    //----------------------------------------------------------------------//
    // Back                                                                 //
    //----------------------------------------------------------------------//
    pPage = GetPage(4);
    pPage->GetPageSize(&sizePage);
    ::SetRect(&SrcRect, 0, 0, sizePage.cx, sizePage.cy);
    GetQFPageRect(4, bPortrait, false, pDC, di, DestRect, bFlip, bRotated);
    ViewMatrix.ScaleAndCenter(DestRect, SrcRect, bFlip);
    pDC->SetViewingMatrix(ViewMatrix);
    pPage->Draw(*pDC);

    //----------------------------------------------------------------------//
    // BatchPrint filename.                                                 //
    //----------------------------------------------------------------------//
    if (pszFileName)
    {
        LOGFONT Font;
        memset(&Font, 0, sizeof(Font));
        Font.lfHeight = -(12 * APP_RESOLUTION / 72);
        lstrcpy(Font.lfFaceName, "Times New Roman");
        pDC->SetFont(&Font);
        pDC->SetTextColor(RGB(0, 0, 0));
        pDC->ExtTextOut(APP_RESOLUTION / 8,
                        SrcRect.bottom - (APP_RESOLUTION / 8), 0, NULL, pszFileName,
                        lstrlen(pszFileName), NULL);
    }

    if (! pDC->EndPage())
    {
        pDC->AbortDoc();
        delete pDC;
        return (false);
    }

    if (! pDC->EndDoc())
    {
        pDC->AbortDoc();
        delete pDC;
        return (false);
    }
    delete pDC;
    return (true);
}


//--------------------------------------------------------------------------//
// Print single-fold card.                                                  //
//--------------------------------------------------------------------------//
bool CAGDoc::PrintCardSingle(PRINTSIDE PrintSide, char *pszDriver, char *pszDevice,
                                char *pszOutput, DEVMODE *pDevMode, bool &bRotated,
                                const char *pszFileName)
{
    if (m_nPages != 4)
        return (false);

    bool bPortrait = (m_AGDocType == DOC_CARDHH || m_AGDocType == DOC_CARDFH);

    if (pDevMode)
    {
        if (bPortrait)
            pDevMode->dmOrientation = DMORIENT_PORTRAIT;
        else
            pDevMode->dmOrientation = DMORIENT_LANDSCAPE;
        pDevMode->dmFields |= DM_ORIENTATION;
    }

    CAGDC *pDC = new CAGDC(pszDriver, pszDevice, pszOutput, pDevMode);
    CAGDCInfo di = pDC->GetDeviceInfo();

    if (pDevMode && pDevMode->dmOrientation == DMORIENT_PORTRAIT &&
        di.m_nHorzSize > di.m_nVertSize)
    {
        delete pDC;
        pDevMode->dmOrientation = DMORIENT_LANDSCAPE;
        pDC = new CAGDC(pszDriver, pszDevice, pszOutput, pDevMode);
        di = pDC->GetDeviceInfo();
    }
    else if (pDevMode && pDevMode->dmOrientation == DMORIENT_LANDSCAPE &&
        di.m_nVertSize > di.m_nHorzSize)
    {
        delete pDC;
        pDevMode->dmOrientation = DMORIENT_PORTRAIT;
        pDC = new CAGDC(pszDriver, pszDevice, pszOutput, pDevMode);
        di = pDC->GetDeviceInfo();
    }

    RECT        DestRect;
    RECT        SrcRect;
    SIZE        sizePage;
    CAGMatrix   ViewMatrix;
    bool        bFlip;

    if (!pDC->StartDoc("Create and Print"))
    {
        delete pDC;
        return (false);
    }

    if (PrintSide == PRINT_OUTSIDE || PrintSide == PRINT_BOTH)
    {
        if (!pDC->StartPage())
        {
            pDC->AbortDoc();
            delete pDC;
            return (false);
        }

        //------------------------------------------------------------------//
        // Front                                                            //
        //------------------------------------------------------------------//
        CAGPage *pPage = GetPage(1);
        pPage->GetPageSize(&sizePage);
        ::SetRect(&SrcRect, 0, 0, sizePage.cx, sizePage.cy);
        GetSFPageRect(1, bPortrait, false, pDC, di, DestRect, bFlip, bRotated);
        ViewMatrix.ScaleAndCenter(DestRect, SrcRect, bFlip);
        pDC->SetViewingMatrix(ViewMatrix);
        pPage->Draw(*pDC);

        //------------------------------------------------------------------//
        // Back                                                             //
        //------------------------------------------------------------------//
        pPage = GetPage(4);
        pPage->GetPageSize(&sizePage);
        ::SetRect(&SrcRect, 0, 0, sizePage.cx, sizePage.cy);
        GetSFPageRect(4, bPortrait, false, pDC, di, DestRect, bFlip, bRotated);
        ViewMatrix.ScaleAndCenter(DestRect, SrcRect, bFlip);
        pDC->SetViewingMatrix(ViewMatrix);
        pPage->Draw(*pDC);

        //------------------------------------------------------------------//
        // BatchPrint filename.                                             //
        //------------------------------------------------------------------//
        if (pszFileName)
        {
            LOGFONT Font;
            memset(&Font, 0, sizeof(Font));
            Font.lfHeight = -(12 * APP_RESOLUTION / 72);
            lstrcpy(Font.lfFaceName, "Times New Roman");
            pDC->SetFont(&Font);
            pDC->SetTextColor(RGB(0, 0, 0));
            pDC->ExtTextOut(APP_RESOLUTION / 8,
                            SrcRect.bottom - (APP_RESOLUTION / 8), 0, NULL, pszFileName,
                            lstrlen(pszFileName), NULL);
        }

        if (!pDC->EndPage())
        {
            pDC->AbortDoc();
            delete pDC;
            return (false);
        }
    }

    if (PrintSide == PRINT_INSIDE || PrintSide == PRINT_BOTH)
    {
        if (!pDC->StartPage())
        {
            pDC->AbortDoc();
            delete pDC;
            return (false);
        }

        bool bPage2Empty = GetPage(2)->IsEmpty();
        bool bPage3Empty = GetPage(3)->IsEmpty();

        //------------------------------------------------------------------//
        // Inside Left / Top                                                //
        //------------------------------------------------------------------//
        CAGPage *pPage = GetPage(2);
        pPage->GetPageSize(&sizePage);
        ::SetRect(&SrcRect, 0, 0, sizePage.cx, sizePage.cy);
        GetSFPageRect(2, bPortrait, !bPage3Empty, pDC, di, DestRect, bFlip,
                        bRotated);
        ViewMatrix.ScaleAndCenter(DestRect, SrcRect, bFlip);
        pDC->SetViewingMatrix(ViewMatrix);
        pPage->Draw(*pDC);

        //------------------------------------------------------------------//
        // BatchPrint filename.                                             //
        //------------------------------------------------------------------//
        if (pszFileName)
        {
            LOGFONT Font;
            memset(&Font, 0, sizeof(Font));
            Font.lfHeight = -(12 * APP_RESOLUTION / 72);
            lstrcpy(Font.lfFaceName, "Times New Roman");
            pDC->SetFont(&Font);
            pDC->SetTextColor(RGB(0, 0, 0));
            pDC->ExtTextOut(APP_RESOLUTION / 8,
                            SrcRect.bottom - (APP_RESOLUTION / 8), 0, NULL, pszFileName,
                            lstrlen(pszFileName), NULL);
        }

        //------------------------------------------------------------------//
        // Inside Right / Bottom                                            //
        //------------------------------------------------------------------//
        pPage = GetPage(3);
        pPage->GetPageSize(&sizePage);
        ::SetRect(&SrcRect, 0, 0, sizePage.cx, sizePage.cy);
        GetSFPageRect(3, bPortrait, ! bPage2Empty, pDC, di, DestRect, bFlip,
                      bRotated);
        ViewMatrix.ScaleAndCenter(DestRect, SrcRect, bFlip);
        pDC->SetViewingMatrix(ViewMatrix);
        pPage->Draw(*pDC);

        if (!pDC->EndPage())
        {
            pDC->AbortDoc();
            delete pDC;
            return (false);
        }
    }

    if (!pDC->EndDoc())
    {
        pDC->AbortDoc();
        delete pDC;
        return (false);
    }
    delete pDC;

    return (true);


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
bool CAGDoc::Read(istream &input)
{
    bool bReturn = true;

    Free();

    CAGDocIO DocIO(&input);

    AGHDR Hdr;
    DocIO.Read(&Hdr, sizeof(Hdr));
    if (memcmp(Hdr.abId, AGDOC_ID, sizeof(Hdr.abId)) == 0 &&
        Hdr.bVer <= AGDOC_VERSION)
    {
        DocIO.Read(&m_AGDocType, sizeof(m_AGDocType));
        DocIO.Read(&m_nPages, sizeof(m_nPages));
        for (int i = 0; i < m_nPages; i++)
        {
            CAGPage *pPage = new CAGPage();
            pPage->Read(&DocIO);
            m_pPages[i] = pPage;
        }
        if (m_nPages > 1)
            m_nCurPage = 1;
    }
    else
        bReturn = false;
    DocIO.Close();

    return bReturn;
}
```

```
//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
bool CAGDoc::Write(ostream &output)
{
    bool bReturn = true;

    CAGDocIO DocIO(&output);

    AGHDR Hdr;
    memcpy(Hdr.abId, AGDOC_ID, sizeof(Hdr.abId));
    Hdr.bVer = AGDOC_VERSION;
    DocIO.Write(&Hdr, sizeof(Hdr));
    DocIO.Write(&m_AGDocType, sizeof(m_AGDocType));
    DocIO.Write(&m_nPages, sizeof(m_nPages));

    for (int i = 0; i < m_nPages; i++)
    {
        CAGPage *pPage = m_pPages[i];
        pPage->Write(&DocIO);
    }
    DocIO.Close();

    return bReturn;
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
CAGDocIO::CAGDocIO(istream *pInput)
{
    m_bInput = true;
    m_bEOF = false;
    m_pInput = pInput;
    m_pOutput = NULL;

    memset(&m_zstream, 0, sizeof(m_zstream));
    m_zstream.next_in = m_zBuf;
    m_zErr = inflateInit2(&m_zstream, -MAX_WBITS);
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
CAGDocIO::CAGDocIO(ostream *pOutput)
{
    m_bInput = false;
    m_pInput = NULL;
    m_pOutput = pOutput;

    memset(&m_zstream, 0, sizeof(m_zstream));
    m_zErr = deflateInit2(&m_zstream, Z_BEST_COMPRESSION, Z_DEFLATED, -MAX_WBITS,
                        DEF_MEM_LEVEL, Z_DEFAULT_STRATEGY);

    m_zstream.next_out = m_zBuf;
    m_zstream.avail_out = Z_BUFSIZE;
}


//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CAGDocIO::Close()
{
    if (m_bInput)
    {
        inflateEnd(&m_zstream);
    }
    else
    {
        bool bDone = false;
        int len = 0;

        m_zstream.avail_in = 0;
        for ( ; ; )
        {
            len = Z_BUFSIZE - m_zstream.avail_out;

            if (len != 0)
            {
                m_pOutput->write((char *)m_zBuf, len);
                m_zstream.next_out = m_zBuf;
                m_zstream.avail_out = Z_BUFSIZE;
            }

            if (bDone)
                break;

            m_zErr = deflate(&m_zstream, Z_FINISH);

            bDone = (m_zstream.avail_out != 0 || m_zErr == Z_STREAM_END);

            if (m_zErr != Z_OK && m_zErr != Z_STREAM_END)
                break;
        }
        deflateEnd(&m_zstream);
    }
}

//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CAGDocIO::Read(void *pData, DWORD dwSize)
{
    if (m_zErr == Z_DATA_ERROR || m_zErr == Z_ERRNO)
        return;
    if (m_zErr == Z_STREAM_END)
        return;

    m_zstream.next_out = (Bytef *)pData;
    m_zstream.avail_out = dwSize;

    while (m_zstream.avail_out != 0)
    {
        if (m_zstream.avail_in == 0 && ! m_bEOF)
        {
            m_pInput->read((char *)m_zBuf, Z_BUFSIZE);

            m_zstream.avail_in = m_pInput->gcount();
            if (m_zstream.avail_in == 0)
                m_bEOF = true;

            m_zstream.next_in = m_zBuf;
        }
        m_zErr = inflate(&m_zstream, Z_NO_FLUSH);

        if (m_zErr != Z_OK || m_bEOF)
            break;
    }
}

//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CAGDocIO::Write(const void *pData, DWORD dwSize)
{
    m_zstream.next_in = (Bytef *)pData;
    m_zstream.avail_in = dwSize;

    while (m_zstream.avail_in != 0)
    {
        if (m_zstream.avail_out == 0)
        {
            m_zstream.next_out = m_zBuf;
            m_pOutput->write((char *)m_zBuf, Z_BUFSIZE);
            m_zstream.avail_out = Z_BUFSIZE;
        }
        deflate(&m_zstream, Z_NO_FLUSH);
    }
}
```

```
#ifndef __AGDIB_H_
#define __AGDIB_H_

inline RGBQUAD *DibColors (const BITMAPINFOHEADER *pbih)
{
    return ((RGBQUAD *) (((BYTE *) pbih) + pbih->biSize +
      (pbih->biCompression == BI_BITFIELDS ? sizeof (DWORD) * 3 : 0)));
}

inline DWORD DibNumColors (const BITMAPINFOHEADER *pbih)
{
    return ((pbih->biClrUsed == 0 && pbih->biBitCount <= 8)
      ? (1 << pbih->biBitCount) : pbih->biClrUsed);
}

inline DWORD DibPaletteSize (const BITMAPINFOHEADER *pbih)
{
    return (DibNumColors (pbih) * sizeof(RGBQUAD));
}

inline BYTE *DibPtr (const BITMAPINFOHEADER *pbih)
{
    return ((BYTE *) (DibColors (pbih) + DibNumColors (pbih)));
}

inline DWORD DibWidthBytes (const BITMAPINFOHEADER *pbih)
{
    return (((pbih->biWidth * pbih->biBitCount + 31L) / 32) * 4);
}

inline DWORD DibSizeImage (const BITMAPINFOHEADER *pbih)
{
    return (pbih->biSizeImage != 0 ? pbih->biSizeImage
      : DibWidthBytes (pbih) * pbih->biHeight);
}

inline DWORD DibSize (const BITMAPINFOHEADER *pbih)
{
    return (pbih->biSize + DibSizeImage (pbih) + DibPaletteSize (pbih) +
      (pbih->biCompression == BI_BITFIELDS ? sizeof (DWORD) * 3 : 0));
}

#endif //__AGDIB_H_
```

```
#ifndef __AGDC_H_
#define __AGDC_H_

#include "AGMatrix.h"
#include "sctypes.h"
#include "scapptyp.h"

typedef DWORD Fixed;
#define IntToFixed(i) (Fixed)(((DWORD)(i)) << 16)
#define FixedToInt(f) (WORD)(((DWORD)f) >> 16)
#define BLTBUFSIZE      65535
#define MAX_LINESPERBLT 20
#define PALETTERGB_FLAG 0x02000000


class CAGDCInfo
{
public:
    CAGDCInfo ();
    void Init (HDC hDC);

public:
    bool    m_bRasDisplay;
    bool    m_bPalette;
    int     m_Technology;
    int     m_nHorzSize;
    int     m_nVertSize;
    int     m_nHorzRes;
    int     m_nVertRes;
    int     m_nLogPixelsX;
    int     m_nLogPixelsY;
    SIZE    m_PhysPageSize;
    SIZE    m_PrintOffset;
};


class CAGDC
{
public:
    CAGDC (const char *pszDriver, const char *pszDevice, const char *pszOutput,
      const DEVMODE *pDevMode);
    CAGDC (HDC hDC);
    virtual ~CAGDC ();

    void DPtoVP (POINT *pPts, int nCount) const
                        { GetDeviceMatrix ().Inverse ().Transform (pPts, nCount); }
    void DPtoVP (RECT *pRect) const
                        { DPtoVP ((POINT *) pRect, 2); }
    void DPAtoVPA (POINT *pPts, int nCount) const
                        { GetDeviceMatrix ().Inverse ().Transform (pPts, nCount, false); }
    void DPAtoVPA (RECT *pRect) const
                        { DPAtoVPA ((POINT *) pRect, 2); }
    void DPtoMP (POINT *pPts, int nCount)
                        { GetVxD ().Inverse ().Transform (pPts, nCount); }
    void DPtoMP (RECT *pRect)
                        { DPtoMP ((POINT *) pRect, 2); }
    void DPAtoMPA (POINT *pPts, int nCount)
                        { GetVxD ().Inverse ().Transform (pPts, nCount, false); }
    void DPAtoMPA (RECT *pRect)
                        { DPAtoMPA ((POINT *) pRect, 2); }
    void DPtoLP (POINT *pPts, int nCount)
                        { GetCTM ().Inverse ().Transform (pPts, nCount); }
    void DPtoLP (RECT *pRect)
                        { DPtoLP ((POINT *) pRect, 2); }
    void DPAtoLPA (POINT *pPts, int nCount)
                        { GetCTM ().Inverse ().Transform (pPts, nCount, false); }
    void DPAtoLPA (RECT *pRect)
                        { DPAtoLPA ((POINT *) pRect, 2); }

    void VPtoMP (POINT *pPts, int nCount) const
                        { GetViewingMatrix ().Inverse ().Transform (pPts, nCount); }
    void VPtoMP (RECT *pRect) const
                        { VPtoMP ((POINT *) pRect, 2); }
    void VPAtoMPA (POINT *pPts, int nCount) const
                        { GetViewingMatrix ().Inverse ().Transform (pPts, nCount, false); }
    void VPAtoMPA (RECT *pRect) const
                        { VPAtoMPA ((POINT *) pRect, 2); }
    void VPtoLP (POINT *pPts, int nCount)
                        { GetMxV ().Inverse ().Transform (pPts, nCount); }
    void VPtoLP (RECT *pRect)
                        { VPtoLP ((POINT *) pRect, 2); }
    void VPAtoLPA (POINT *pPts, int nCount)
                        { GetMxV ().Inverse ().Transform (pPts, nCount, false); }
    void VPAtoLPA (RECT *pRect)
                        { VPAtoLPA ((POINT *) pRect, 2); }

    void MPtoLP (POINT *pPts, int nCount) const
                        { GetModelingMatrix ().Inverse ().Transform (pPts, nCount); }
    void MPtoLP (RECT *pRect) const
                        { MPtoLP ((POINT *) pRect, 2); }
    void MPAtoLPA (POINT *pPts, int nCount) const
                        { GetModelingMatrix ().Inverse ().Transform (pPts, nCount, false); }
    void MPAtoLPA (RECT *pRect) const
                        { MPAtoLPA ((POINT *) pRect, 2); }

    void LPtoMP (POINT *pPts, int nCount) const
                        { GetModelingMatrix ().Transform (pPts, nCount); }
    void LPtoMP (RECT *pRect) const
                        { LPtoMP ((POINT *) pRect, 2); }
    void LPAtoMPA (POINT *pPts, int nCount) const
                        { GetModelingMatrix ().Transform (pPts, nCount, false); }
    void LPAtoMPA (RECT *pRect) const
                        { LPAtoMPA ((POINT *) pRect, 2); }
    void LPtoVP (POINT *pPts, int nCount)
                        { GetMxV ().Transform (pPts, nCount); }
    void LPtoVP (RECT *pRect)
                        { LPtoVP ((POINT *) pRect, 2); }
    void LPAtoVPA (POINT *pPts, int nCount)
                        { GetMxV ().Transform (pPts, nCount, false); }
    void LPAtoVPA (RECT *pRect)
                        { LPAtoVPA ((POINT *) pRect, 2); }
    void LPtoDP (POINT *pPts, int nCount)
                        { GetCTM ().Transform (pPts, nCount); }
    void LPtoDP (RECT *pRect)
                        { LPtoDP ((POINT *) pRect, 2); }
    void LPAtoDPA (POINT *pPts, int nCount)
                        { GetCTM ().Transform (pPts, nCount, false); }
    void LPAtoDPA (RECT *pRect)
                        { LPAtoDPA ((POINT *) pRect, 2); }

    void MPtoVP (POINT *pPts, int nCount) const
                        { GetViewingMatrix ().Transform (pPts, nCount); }
    void MPtoVP (RECT *pRect) const
                        { MPtoVP ((POINT *) pRect, 2); }
    void MPAtoVPA (POINT *pPts, int nCount) const
                        { GetViewingMatrix ().Transform (pPts, nCount, false); }
    void MPAtoVPA (RECT *pRect) const
                        { MPAtoVPA ((POINT *) pRect, 2); }
    void MPtoDP (POINT *pPts, int nCount)
                        { GetVxD ().Transform (pPts, nCount); }
    void MPtoDP (RECT *pRect)
                        { MPtoDP ((POINT *) pRect, 2); }
    void MPAtoDPA (POINT *pPts, int nCount)
                        { GetVxD ().Transform (pPts, nCount, false); }
    void MPAtoDPA (RECT *pRect)
                        { MPAtoDPA ((POINT *) pRect, 2); }

    void VPtoDP (POINT *pPts, int nCount) const
                        { GetDeviceMatrix ().Transform (pPts, nCount); }
    void VPtoDP (RECT *pRect) const
                        { VPtoDP ((POINT *) pRect, 2); }
    void VPAtoDPA (POINT *pPts, int nCount) const
                        { GetDeviceMatrix ().Transform (pPts, nCount, false); }
    void VPAtoDPA (RECT *pRect) const
                        { VPAtoDPA ((POINT *) pRect, 2); }
    bool AbortDoc () const
                        { return (::AbortDoc (m_hDC) >= 0); }
    bool EndDoc () const
                        { return (::EndDoc (m_hDC) >= 0); }
    bool EndPage () const
                        { return (::EndPage (m_hDC) >= 0); }
    void ExtTextOut (int x, int y, UINT nOptions, const RECT *pRect,
                        const TCHAR *pszString, UINT nCount, const int *pDxWidths);
    static void Free ();
    const CAGMatrix &GetCTM ();
    HDC GetHDC () const
                        { return (m_hDC); }
    const CAGDCInfo &GetDeviceInfo () const
                        { return (m_Info); }
    const CAGMatrix &GetDeviceMatrix () const
                        { return (m_MatrixD); }
    int GetMinAWidth (UINT uFirstChar, UINT uLastChar);
    void GetTextMetrics (TEXTMETRIC *ptm) const;
    HWND GetWnd () const
                        { return (m_hWnd); }
    const CAGMatrix &GetModelingMatrix () const
                        { return (m_MatrixM); }
    const CAGMatrix &GetMxV ();
    void GetTextExtent (const TCHAR *pString, int nCount, SIZE *pSize);
    const CAGMatrix &GetViewingMatrix () const
                        { return (m_MatrixV); }
    const CAGMatrix &GetVxD ();
    void InvertLine (POINT ptFrom, POINT ptTo);
    void InvertRect (const RECT &Rect);
    void Polygon (const POINT *pPts, int nPoints);
    void PopModelingMatrix ()
                        { m_MatrixM.Unity (); m_bUpdateMxV = true; m_bUpdateCTM = true; }
    void PushModelingMatrix (const CAGMatrix &Matrix)
                        { m_MatrixM = Matrix; m_bUpdateMxV = true; m_bUpdateCTM = true; }
    void Rectangle (const RECT &Rect);
    void SetDeviceMatrix (CAGMatrix &Matrix)
                        { m_MatrixD = Matrix; m_bUpdateVxD = true; m_bUpdateCTM = true; }
    void SetFont (const UFont &Font);
    void SetTextColor (COLORREF Color) const;
    bool SetTransformMode (bool bOn)
                        { bool bCur = m_bDoTransform; m_bDoTransform = bOn; return (bCur); }
    void SetViewingMatrix (CAGMatrix &Matrix)
                        { m_MatrixV = Matrix; m_bUpdateVxD = true; m_bUpdateMxV = true;
                        m_bUpdateCTM = true; }
    bool StartDoc (const char *pszDocName) const;
    bool StartPage ();
    void StretchBlt (RECT DestRect, const void *pvBits, const BITMAPINFO *pbi);

protected:
    CAGDC ()
        {  }
    void CleanUp ();
    void CreatePalette ();
    void Dither (const BYTE *pSrcBits, BYTE *pDstBits, int nSrcBitCount,
    int nDstWidth, int y, Fixed fixSrcStepX, const RGBQUAD *pColors) const;
    void FlipDIB (const BYTE *pBits, BYTE *pNewBits,
    const BITMAPINFOHEADER *pbih, bool bFlipX, bool bFlipY) const;
    void StretchBlt2 (int nDstX, int nDstY, int nDstWidth, int nDstHeight,
    const void *pvBits, const BITMAPINFO *pbi) const;
    Fixed FixedDivide (Fixed Dividend, Fixed Divisor) const;
    void Init ();

protected:
    bool            m_bGivenDC;
    HDC             m_hDC;
    HWND            m_hWnd;
    static BYTE     *m_pBltBuf;
    static HPALETTE m_hPalette;
    HPALETTE        m_hOldPalette;
    CAGDCInfo       m_Info;
    CAGMatrix       m_MatrixM;
    CAGMatrix       m_MatrixV;
    CAGMatrix        m_MatrixD;
    CAGMatrix        m_MatrixVxD;
    CAGMatrix        m_MatrixMxV;
    CAGMatrix        m_MatrixCTM;
    bool             m_bUpdateVxD;
    bool             m_bUpdateMxV;
    bool             m_bUpdateCTM;
    bool             m_bDoTransform;
    HFONT            m_hOldFont;
    UFont            m_CurFont;
};


class CAGPaintDC : public CAGDC
{
public:
    CAGPaintDC (HWND hWnd);
    ~CAGPaintDC ();

protected:
    PAINTSTRUCT m_PaintStruct;
};


class CAGClientDC : public CAGDC
{
public:
    CAGClientDC (HWND hWnd);
    ~CAGClientDC ();
};


class CAGIC : public CAGDC
{
public:
    CAGIC (const char *pszDriver, const char *pszDevice = NULL,
      const char *pszOutput = NULL, const DEVMODE *pDevMode = NULL);
};


class CAGDIBSectionDC : public CAGDC
{
public:
    CAGDIBSectionDC (const BITMAPINFO *pbmi, UINT iUsage, BYTE **ppvBits);
    ~CAGDIBSectionDC ();

protected:
    HBITMAP m_hBitmap;
    HBITMAP m_hOldBitmap;
};

#endif //__AGDC_H_
```

```
//==========================================================================//
//==========================================================================//
#include "stdafx.h"
#include "AGDC.h"
#include "AGDib.h"

#ifdef _AFX
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif
#endif

BYTE *CAGDC::m_pBltBuf = NULL;
HPALETTE CAGDC::m_hPalette = NULL;

static const BYTE aHalftone16x16[16][16] =
{
     0, 44,  9, 41,  3, 46, 12, 43,  1, 44, 10, 41,  3, 46, 12, 43,
    34, 16, 25, 19, 37, 18, 28, 21, 35, 16, 26, 19, 37, 18, 28, 21,
    38,  6, 47,  3, 40,  9, 50,  6, 38,  7, 47,  4, 40,  9, 49,  6,
    22, 28, 13, 31, 25, 31, 15, 34, 22, 29, 13, 32, 24, 31, 15, 34,
     2, 46, 12, 43,  1, 45, 10, 42,  2, 45, 11, 42,  1, 45, 11, 42,
    37, 18, 27, 21, 35, 17, 26, 20, 36, 17, 27, 20, 36, 17, 26, 20,
    40,  8, 49,  5, 38,  7, 48,  4, 39,  8, 48,  5, 39,  7, 48,  4,
    24, 30, 15, 33, 23, 29, 13, 32, 23, 30, 14, 33, 23, 29, 14, 32,
     2, 46, 12, 43,  0, 44, 10, 41,  3, 47, 12, 44,  0, 44, 10, 41,
    37, 18, 27, 21, 35, 16, 25, 19, 37, 19, 28, 22, 35, 16, 25, 19,
    40,  9, 49,  5, 38,  7, 47,  4, 40,  9, 50,  6, 38,  6, 47,  3,
    24, 30, 15, 34, 22, 29, 13, 32, 25, 31, 15, 34, 22, 28, 13, 31,
     1, 45, 11, 42,  2, 46, 11, 42,  1, 45, 10, 41,  2, 46, 11, 43,
    36, 17, 26, 20, 36, 17, 27, 21, 35, 16, 26, 20, 36, 18, 27, 21,
    39,  8, 48,  4, 39,  8, 49,  5, 38,  7, 48,  4, 39,  8, 49,  5,
    23, 29, 14, 33, 24, 30, 14, 33, 23, 29, 13, 32, 24, 30, 14, 33
};

static const BYTE aModulo51[256] =
{
     0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
24, 25,
    26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49,
50,
     0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
24, 25,
    26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49,
50,
     0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
24, 25,
    26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49,
50,
     0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
24, 25,
    26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49,
50,
     0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
24, 25,
    26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49,
50,
     0
};

static const BYTE aDividedBy51[256] =
{
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
     1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
     2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
     2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
     3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
     3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
     4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
    4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
    5
};

static const BYTE aTimes6[6] =
{
  0, 6, 12, 18, 24, 30
};

static const BYTE aTimes36[6] =
{
  0, 36, 72, 108, 144, 180
};


//-------------------------------------------------------------------------//
//-------------------------------------------------------------------------//
CAGDCInfo::CAGDCInfo ()
{
    m_nHorzSize = 0;
    m_nVertSize = 0;
    m_nHorzRes = 0;
    m_nVertRes = 0;
    m_nLogPixelsX = 0;
    m_nLogPixelsY = 0;
    m_PhysPageSize.cx = m_PhysPageSize.cy = 0;
    m_PrintOffset.cx = m_PrintOffset.cy = 0;
}

//-------------------------------------------------------------------------//
//-------------------------------------------------------------------------//
void CAGDCInfo::Init (HDC hDC)
{
    m_Technology = ::GetDeviceCaps (hDC, TECHNOLOGY);
    m_bRasDisplay = (m_Technology == DT_RASDISPLAY);
    m_bPalette = ((::GetDeviceCaps (hDC, RASTERCAPS) & RC_PALETTE) != 0);

    m_nHorzSize = ::GetDeviceCaps (hDC, HORZSIZE);
    m_nVertSize = ::GetDeviceCaps (hDC, VERTSIZE);
    m_nHorzRes = ::GetDeviceCaps (hDC, HORZRES);
    m_nVertRes = ::GetDeviceCaps (hDC, VERTRES);
    m_nLogPixelsX = ::GetDeviceCaps (hDC, LOGPIXELSX);
    m_nLogPixelsY = ::GetDeviceCaps (hDC, LOGPIXELSY);

    if (::Escape (hDC, GETPHYSPAGESIZE, NULL, NULL, &m_PhysPageSize) <= 0)
    {
        m_PhysPageSize.cx = m_nHorzRes;
        m_PhysPageSize.cy = m_nVertRes;
    }
    if (::Escape (hDC, GETPRINTINGOFFSET, NULL, NULL, &m_PrintOffset) <= 0)
        m_PrintOffset.cx = m_PrintOffset.cy = 0;
}


//-------------------------------------------------------------------------//
//-------------------------------------------------------------------------//
CAGDC::CAGDC (const char *pszDriver, const char *pszDevice,
  const char *pszOutput, const DEVMODE *pDevMode)
{
    m_hDC = ::CreateDC (pszDriver, pszDevice, pszOutput, pDevMode);
    Init ();
}

//-------------------------------------------------------------------------//
//-------------------------------------------------------------------------//
CAGDC::CAGDC (HDC hDC)
{
    m_hDC = hDC;
    Init ();
    m_bGivenDC = true;
}

//-------------------------------------------------------------------------//
```

```
//---------------------------------------------------------------------------//
CAGDC::~CAGDC ()
{
    if (m_hDC)
    {
        CleanUp ();
        if (! m_bGivenDC)
            ::DeleteDC (m_hDC);
        m_hDC = NULL;
    }
}

//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
void CAGDC::CleanUp ()
{
    if (m_hOldPalette)
        ::SelectPalette (m_hDC, m_hOldPalette, true);
    if (m_hOldFont)
    {
        HFONT hFont = (HFONT) ::SelectObject (m_hDC, m_hOldFont);
        ::DeleteObject (hFont);
    }
}

//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
void CAGDC::CreatePalette ()
{
    m_hPalette = NULL;
    LPLOGPALETTE lpLogPalette;

    int nPaletteSize = sizeof (LOGPALETTE) + (216 * sizeof (PALETTEENTRY));
    if ((lpLogPalette = (LPLOGPALETTE) malloc (nPaletteSize)) != NULL)
    {
        memset (lpLogPalette, 0, nPaletteSize);
        lpLogPalette->palVersion = 0x300;
        lpLogPalette->palNumEntries = 216;
        for (int r = 0, n = 0; r < 6; r++)
        {
            for (int g = 0; g < 6; g++)
            {
                for (int b = 0; b < 6; b++, n++)
                {
                    lpLogPalette->palPalEntry[n].peRed = (BYTE) (r * 51);
                    lpLogPalette->palPalEntry[n].peGreen = (BYTE) (g * 51);
                    lpLogPalette->palPalEntry[n].peBlue = (BYTE) (b * 51);
                }
            }
        }
        m_hPalette = ::CreatePalette (lpLogPalette);
        free (lpLogPalette);
    }
}

//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
void CAGDC::Dither (const BYTE *pSrcBits, BYTE *pDstBits, int nSrcBitCount,
  int nDstWidth, int y, Fixed fixSrcStepX, const RGBQUAD *pColors) const
{
    Fixed fixPosX = (fixSrcStepX >> 1);
    WORD wSrcPosX = 0;
    int Red = 0;
    int Green = 0;
    int Blue = 0;
    BYTE bIndex, bTemp, bRedTemp, bGreenTemp, bBlueTemp;

    for (int x = 0; x < nDstWidth; x++)
    {
        wSrcPosX = FixedToInt (fixPosX);

        if (nSrcBitCount == 8)
        {
            bIndex = pSrcBits[wSrcPosX];
            Red = pColors[bIndex].rgbRed;
            Green = pColors[bIndex].rgbGreen;
            Blue = pColors[bIndex].rgbBlue;
        }
        else if (nSrcBitCount == 24)
        {
            wSrcPosX *= 3;
            Blue = pSrcBits[wSrcPosX];
            Green = pSrcBits[wSrcPosX + 1];
            Red = pSrcBits[wSrcPosX + 2];
        }
        else if (nSrcBitCount == 4)
        {
            if (wSrcPosX % 2)
                bIndex = (BYTE) (pSrcBits[wSrcPosX >> 1] & 0x0f);
            else
                bIndex = (BYTE) (pSrcBits[wSrcPosX >> 1] >> 4);
            Red = pColors[bIndex].rgbRed;
            Green = pColors[bIndex].rgbGreen;
            Blue = pColors[bIndex].rgbBlue;
        }

        fixPosX += fixSrcStepX;

        bTemp = aHalftone16x16[(x & 0x0f)][(y & 0x0f)];
        bRedTemp = aDividedBy51[Red];
        bGreenTemp = aDividedBy51[Green];
        bBlueTemp = aDividedBy51[Blue];

        if (aModulo51[Red] > bTemp)
            bRedTemp++;
        if (aModulo51[Green] > bTemp)
            bGreenTemp++;
        if (aModulo51[Blue] > bTemp)
            bBlueTemp++;

        pDstBits[x] = (BYTE) (aTimes36[bRedTemp] + aTimes6[bGreenTemp] + bBlueTemp);
    }
}

//--------------------------------------------------------------------------//
//--------------------------------------------------------------------------//
void CAGDC::ExtTextOut( int x, int y, UINT nOptions, const RECT *pRect,
  const TCHAR *pszString, UINT nCount, const int *pDxWidths)
{
    if (m_bDoTransform)
    {
        int *pWidths = NULL;
        if (pDxWidths)
        {
            pWidths = new int [nCount];

            CAGMatrix CTM = GetCTM ();
            int Angle = 0;
            if (CTM.GetRotation (Angle))
                CTM.Rotate (-Angle / 10);

            int nCurX = 0;
            int nPrevX = 0;
            for (UINT i = 0; i < nCount; i++)
            {
                nCurX += pDxWidths[i];
                POINT Pt = {nCurX, 0};

                if (Angle != 0)
                    CTM.Transform (&Pt, 1, false);
                else
                    LPAtoDPA (&Pt, 1);

                pWidths[i] = Pt.x - nPrevX;
                nPrevX = Pt.x;
            }
        }

        POINT Pt = {x, y};
        LPtoDP (&Pt, 1);

        ::ExtTextOut (m_hDC, Pt.x, Pt.y, nOptions, pRect, pszString, nCount,
          pWidths);

        if (pWidths)
            delete [] pWidths;
    }
    else
    {
        ::ExtTextOut (m_hDC, x, y, nOptions, pRect, pszString, nCount,
          pDxWidths);
    }
}

//------------------------------------------------------------------------------//
// Fixed point division.                                                        //
//------------------------------------------------------------------------------//
Fixed CAGDC::FixedDivide (Fixed Dividend, Fixed Divisor) const
{
    Fixed fixResult = 0;
    _asm
    {
        mov     eax,Dividend
        mov     ecx,Divisor

        rol     eax,10h
        movsx   edx,ax
        xor     ax,ax

        idiv    ecx
        shld    edx,eax,16
        mov     [fixResult], eax
    };
    return (fixResult);
}

//------------------------------------------------------------------------------//
// Flip a DIB horizontally, vertically or both.                                 //
//                                                                              //
// Currently only supports 8 and 24 bit DIBs.                                   //
//------------------------------------------------------------------------------//
void CAGDC::FlipDIB (const BYTE *pBits, BYTE *pNewBits,
  const BITMAPINFOHEADER *pbih, bool bFlipX, bool bFlipY) const
{
    const BYTE *pSrc = pBits;
    int nWidth = DibWidthBytes (pbih);

    BYTE *pDest;
    int nDestInc;

    if (bFlipY)
    {
        pDest = pNewBits + (nWidth * (pbih->biHeight - 1));
        nDestInc = -nWidth;
    }
    else
    {
        pDest = pNewBits;
        nDestInc = nWidth;
    }

    for (int y = 0; y < pbih->biHeight; y++)
    {
        if (bFlipX)
        {
            if (pbih->biBitCount == 8)
            {
                for (int x = 0; x < pbih->biWidth; x++)
                    pDest[x] = pSrc[pbih->biWidth - x - 1];
            }
            else if (pbih->biBitCount == 24)
            {
                for (int x = 0; x < pbih->biWidth; x++)
                {
                    int nDstOffset = x * 3;
                    int nSrcOffset = (pbih->biWidth - x - 1) * 3;
                    pDest[nDstOffset] = pSrc[nSrcOffset];
                    pDest[nDstOffset + 1] = pSrc[nSrcOffset + 1];
                    pDest[nDstOffset + 2] = pSrc[nSrcOffset + 2];
                }
            }
        }
        else
            memcpy (pDest, pSrc, nWidth);

        pSrc += nWidth;
        pDest += nDestInc;
    }
}

//------------------------------------------------------------------------//
//------------------------------------------------------------------------//
void CAGDC::Free ()
{
    if (m_hPalette)
    {
        ::DeleteObject (m_hPalette);
        m_hPalette = NULL;
    }

    if (m_pBltBuf)
    {
        free (m_pBltBuf);
        m_pBltBuf = NULL;
    }
}

//------------------------------------------------------------------------//
//------------------------------------------------------------------------//
const CAGMatrix &CAGDC::GetCTM ()
{
    if (m_bUpdateCTM)
    {
        m_MatrixCTM = GetModelingMatrix () * GetVxD ();
        m_bUpdateCTM = false;
    }
    return (m_MatrixCTM);
}

//------------------------------------------------------------------------//
//------------------------------------------------------------------------//
int CAGDC::GetMinAWidth (UINT uFirstChar, UINT uLastChar)
{
    POINT   MinAWidth = {0, 0};
    int     nChars = uLastChar - uFirstChar + 1;
    ABC     *pABC = new ABC [nChars];

    if (::GetCharABCWidths (m_hDC, uFirstChar, uLastChar, pABC))
    {
        for (int i = 0; i < nChars; i++)
            MinAWidth.x = min (MinAWidth.x, pABC[i].abcA);
    }
    delete [] pABC;

    if (m_bDoTransform)
        LPAtoDPA (&MinAWidth, 1);

    return (MinAWidth.x);
}

//------------------------------------------------------------------------//
//------------------------------------------------------------------------//
```

```
const CAGMatrix &CAGDC::GetVxD ()
{
    if (m_bUpdateVxD)
    {
        m_MatrixVxD = GetViewingMatrix () * GetDeviceMatrix ();
        m_bUpdateVxD = false;
    }
    return (m_MatrixVxD);
}

//----------------------------------------------------------------------------//
//----------------------------------------------------------------------------//
const CAGMatrix &CAGDC::GetMxV ()
{
    if (m_bUpdateMxV)
    {
        m_MatrixVxD = GetModelingMatrix () * GetViewingMatrix ();
        m_bUpdateMxV = false;
    }
    return (m_MatrixMxV);
}


//----------------------------------------------------------------------------//
//----------------------------------------------------------------------------//
void CAGDC::GetTextExtent (const TCHAR *pString, int nCount, SIZE *pSize)
{
    ::GetTextExtentPoint32 (m_hDC, pString, nCount, pSize);
    if (m_bDoTransform)
        LPAtoDPA ((POINT *) pSize, 1);
}

//----------------------------------------------------------------------------//
//----------------------------------------------------------------------------//
void CAGDC::GetTextMetrics (TEXTMETRIC *ptm) const
{
    ::GetTextMetrics (m_hDC, ptm);
    if (m_bDoTransform)
    {
        // todo
    }
}

//----------------------------------------------------------------------------//
//----------------------------------------------------------------------------//
void CAGDC::Init ()
{
    m_Info.Init (m_hDC);

    m_bGivenDC = false;
    m_hWnd = NULL;
    m_hOldFont = NULL;
    m_hOldPalette = NULL;
    if (m_Info.m_bPalette)
    {
        if (m_hPalette == NULL)
            CreatePalette ();

        m_hOldPalette = ::SelectPalette (m_hDC, m_hPalette, true);
        ::RealizePalette (m_hDC);
    }

    ::SetMapMode (m_hDC, MM_TEXT);
    ::SetTextAlign (m_hDC, TA_LEFT | TA_BASELINE | TA_NOUPDATECP);
    ::SetBkMode (m_hDC, TRANSPARENT);

    m_MatrixD.SetMatrix ((double) m_Info.m_nLogPixelsX / (double) APP_RESOLUTION,
      0, 0, (double) m_Info.m_nLogPixelsY / (double) APP_RESOLUTION,
      -m_Info.m_PrintOffset.cx, -m_Info.m_PrintOffset.cy);

    m_bUpdateVxD = true;
    m_bUpdateMxV = false;
    m_bUpdateCTM = true;
    m_bDoTransform = true;
}

//------------------------------------------------------------------------//
//------------------------------------------------------------------------//
void CAGDC::InvertLine (POINT ptFrom, POINT ptTo)
{
    HPEN hOldPen = (HPEN) ::SelectObject (m_hDC, ::GetStockObject (BLACK_PEN));
    int OldROP2 = ::SetROP2 (m_hDC, R2_NOT);

    if (m_bDoTransform)
    {
        LPtoDP (&ptFrom, 1);
        LPtoDP (&ptTo, 1);
    }

    ::MoveToEx (m_hDC, ptFrom.x, ptFrom.y, NULL);
    ::LineTo (m_hDC, ptTo.x, ptTo.y);

    ::SetROP2 (m_hDC, OldROP2);
    ::SelectObject (m_hDC, hOldPen);
}

//------------------------------------------------------------------------//
//------------------------------------------------------------------------//
void CAGDC::InvertRect (const RECT &Rect)
{
    HBRUSH hOldBrush = (HBRUSH) ::SelectObject (m_hDC,
      ::GetStockObject (BLACK_BRUSH));
    HPEN hOldPen = (HPEN) ::SelectObject (m_hDC, ::GetStockObject (NULL_PEN));
    int OldROP2 = ::SetROP2 (m_hDC, R2_NOT);

    Rectangle (Rect);

    ::SetROP2 (m_hDC, OldROP2);
    ::SelectObject (m_hDC, hOldPen);
    ::SelectObject (m_hDC, hOldBrush);
}

//------------------------------------------------------------------------//
//------------------------------------------------------------------------//
void CAGDC::Polygon (const POINT *pPts, int nPoints)
{
    POINT *pPoints = new POINT [nPoints];
    memcpy (pPoints, pPts, nPoints * sizeof (POINT));

    if (m_bDoTransform)
        LPtoDP (pPoints, nPoints);

    ::Polygon (m_hDC, pPoints, nPoints);

    delete [] pPoints;
}

//------------------------------------------------------------------------//
//------------------------------------------------------------------------//
void CAGDC::Rectangle (const RECT &Rect)
{
    RECT r = Rect;

    if (m_bDoTransform)
    {
        int Angle = 0;
        if (GetCTM ().GetRotation (Angle))
        {
            POINT Pts[4];
            Pts[0].x = Pts[1].x = r.left;
            Pts[2].x = Pts[3].x = r.right;
            Pts[0].y = Pts[3].y = r.top;
            Pts[1].y = Pts[2].y = r.bottom;

            Polygon (Pts, 4);
            return;
            }
            LPtoDP (&r);
        }

        ::Rectangle (m_hDC, r.left, r.top, r.right, r.bottom);
}

//-------------------------------------------------------------------------//
//-------------------------------------------------------------------------//
void CAGDC::SetFont (const UFont &Font)
{
    LOGFONT lf = Font;

    if (m_bDoTransform)
    {
        POINT Pt = { lf.lfWidth, abs (lf.lfHeight) };

        CAGMatrix CTM = GetCTM ();
        int Angle = 0;
        if (CTM.GetRotation (Angle))
        {
            CTM.Rotate (-Angle / 10);
            CTM.Transform (&Pt, 1, false);
            lf.lfEscapement = Angle;
        }
        else
            LPAtoDPA (&Pt, 1);

        if (Font.lfWidth != 0)
            lf.lfWidth = Pt.x;
        lf.lfHeight = (Font.lfHeight < 0) ? -abs(Pt.y) : abs(Pt.y);
    }

    if (m_CurFont != lf)
    {
        HFONT hFont = ::CreateFontIndirect (&lf);
        HFONT hOldFont = (HFONT) ::SelectObject (m_hDC, hFont);
        if (m_hOldFont)
            ::DeleteObject (hOldFont);
        else
            m_hOldFont = hOldFont;
        m_CurFont = lf;
    }
}

//-------------------------------------------------------------------------//
//-------------------------------------------------------------------------//
void CAGDC::SetTextColor (COLORREF Color) const
{
    if (m_Info.m_bPalette)
        Color |= PALETTERGB_FLAG;

    ::SetTextColor (m_hDC, Color);
}

//-------------------------------------------------------------------------//
//-------------------------------------------------------------------------//
bool CAGDC::StartDoc (const char *pszDocName) const
{
    DOCINFO di;
    memset (&di, 0, sizeof (di));
    di.cbSize = sizeof (di);
    di.lpszDocName = pszDocName;
    return (::StartDoc (m_hDC, &di) > 0);
}

//-------------------------------------------------------------------------//
//-------------------------------------------------------------------------//
bool CAGDC::StartPage ()
{
    if (m_hOldFont)
    {
        HFONT hFont = (HFONT) ::SelectObject (m_hDC, m_hOldFont);
```

```
            ::DeleteObject (hFont);
            m_hOldFont = NULL;

            LOGFONT lf;
            memset (&lf, 0, sizeof (lf));
            m_CurFont.setLogFont (lf);
        }

    bool bReturn = (::StartPage (m_hDC) > 0);

    ::SetMapMode (m_hDC, MM_TEXT);
    ::SetTextAlign (m_hDC, TA_LEFT | TA_BASELINE | TA_NOUPDATECP);
    ::SetBkMode (m_hDC, TRANSPARENT);

    return (bReturn);
}

//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
void CAGDC::StretchBlt (RECT DestRect, const void *pvBits,
  const BITMAPINFO *pbi)
{
    if (m_bDoTransform)
        LPtoDP (&DestRect);

    if (WIDTH (DestRect) == 0 || HEIGHT (DestRect) == 0)
        return;

    if (m_Info.m_bRasDisplay)
    {
        if (m_pBltBuf == NULL)
            m_pBltBuf = (BYTE *) malloc (BLTBUFSIZE);

        if (m_pBltBuf)
        {
            StretchBlt2 (DestRect.left, DestRect.top,
              WIDTH (DestRect), HEIGHT (DestRect), pvBits, pbi);
        }
    }
    else
    {
        bool bFlipX = (WIDTH (DestRect) < 0);
        bool bFlipY = (HEIGHT (DestRect) < 0);
        if (bFlipX || bFlipY)
        {
            BYTE *pNewBits = (BYTE *) malloc (DibSizeImage (&pbi->bmiHeader));
            FlipDIB ((BYTE *) pvBits, pNewBits, &pbi->bmiHeader, bFlipX, bFlipY);
            if (bFlipX)
                SWAP (DestRect.left, DestRect.right);
            if (bFlipY)
                SWAP (DestRect.top, DestRect.bottom);
            ::StretchDIBits (m_hDC, DestRect.left, DestRect.top,
              WIDTH (DestRect), HEIGHT (DestRect),
              0, 0, pbi->bmiHeader.biWidth, pbi->bmiHeader.biHeight,
              pNewBits, pbi, DIB_RGB_COLORS, SRCCOPY);
            free (pNewBits);
        }
        else
        {
            ::StretchDIBits (m_hDC, DestRect.left, DestRect.top,
              WIDTH (DestRect), HEIGHT (DestRect),
              0, 0, pbi->bmiHeader.biWidth, pbi->bmiHeader.biHeight,
              pvBits, pbi, DIB_RGB_COLORS, SRCCOPY);
        }
    }
}

//---------------------------------------------------------------------------//
//---------------------------------------------------------------------------//
void CAGDC::StretchBlt2 (int nDstX, int nDstY, int nDstWidth, int nDstHeight,
  const void *pvBits, const BITMAPINFO *pbi) const
{
    struct
    {
        BITMAPINFOHEADER    bih;
        RGBQUAD             Colors[256];
    } NewHdr;

    NewHdr.bih.biSize = sizeof (BITMAPINFOHEADER);
    NewHdr.bih.biWidth = nDstWidth;
    NewHdr.bih.biHeight = 0;
    NewHdr.bih.biPlanes = 1;
    NewHdr.bih.biCompression = BI_RGB;
    NewHdr.bih.biSizeImage = 0;
    NewHdr.bih.biXPelsPerMeter = 0;
    NewHdr.bih.biYPelsPerMeter = 0;
    NewHdr.bih.biClrUsed = 0;
    NewHdr.bih.biClrImportant = 0;

    UINT iUsage = DIB_RGB_COLORS;

    if (m_Info.m_bPalette)
    {
        NewHdr.bih.biBitCount = 8;
        NewHdr.bih.biClrUsed = 256;
        for (WORD i = 0; i < 256; i++)
            ((WORD *) NewHdr.Colors)[i] = i;
        iUsage = DIB_PAL_COLORS;
    }
    else
    {
        if (pbi->bmiHeader.biBitCount == 1 || pbi->bmiHeader.biBitCount == 4)
            NewHdr.bih.biBitCount = 8;
        else
            NewHdr.bih.biBitCount = pbi->bmiHeader.biBitCount;

        NewHdr.bih.biClrUsed = pbi->bmiHeader.biClrUsed;
        memcpy (NewHdr.Colors, pbi->bmiColors, pbi->bmiHeader.biClrUsed * sizeof (RGBQUAD));
    }

    DWORD dwSrcWidthBytes = DibWidthBytes (&pbi->bmiHeader);
    DWORD dwDstWidthBytes = DibWidthBytes (&NewHdr.bih);
    int nMaxLinesPerBlt = BLTBUFSIZE / dwDstWidthBytes;
    if (nMaxLinesPerBlt > MAX_LINESPERBLT)
        nMaxLinesPerBlt = MAX_LINESPERBLT;
    NewHdr.bih.biHeight = nMaxLinesPerBlt;

    BYTE *pSrcBitsStart = ((BYTE *) pvBits) + ((pbi->bmiHeader.biHeight - 1) * dwSrcWidthBytes);
    BYTE *pSrcBits = NULL;
    BYTE *pDstBits = m_pBltBuf + ((nMaxLinesPerBlt - 1) * dwDstWidthBytes);
    BYTE *pPrevDstBits = NULL;

    bool bNoXStretch = (nDstWidth == pbi->bmiHeader.biWidth && pbi->bmiHeader.biBitCount != 4
      && pbi->bmiHeader.biBitCount != 1);
    int nLinesToBlt = 0;

    Fixed fixSrcStepX = FixedDivide (IntToFixed (pbi->bmiHeader.biWidth), IntToFixed (nDstWidth));
    Fixed fixSrcStepY = FixedDivide (IntToFixed (pbi->bmiHeader.biHeight), IntToFixed (nDstHeight));
    Fixed fixPosX = 0;
    Fixed fixPosY = (fixSrcStepY >> 1);
    WORD wSrcPosX = 0;
    WORD wSrcPosY = 0;
    WORD wPrevPosY = 0xffff;

    for (int y = 0; y < nDstHeight; y++)
    {
        wSrcPosY = FixedToInt (fixPosY);
        pSrcBits = pSrcBitsStart - (wSrcPosY * dwSrcWidthBytes);

        if (m_Info.m_bPalette)
        {
            Dither (pSrcBits, pDstBits, pbi->bmiHeader.biBitCount, nDstWidth, y, fixSrcStepX,
              pbi->bmiColors);
        }
        else if (wSrcPosY == wPrevPosY)
            memcpy (pDstBits, pPrevDstBits, dwDstWidthBytes);
```

```
          0, 0, nDstWidth, nLinesToBlt, pDstBits, (BITMAPINFO *) &NewHdr.bih,
          iUsage, SRCCOPY);
    }
}

//------------------------------------------------------------------------------//
//------------------------------------------------------------------------------//
CAGPaintDC::CAGPaintDC (HWND hWnd)
{
    m_hDC = ::BeginPaint (hWnd, &m_PaintStruct);
    Init ();
    m_hWnd = hWnd;
}

//------------------------------------------------------------------------------//
//------------------------------------------------------------------------------//
CAGPaintDC::~CAGPaintDC ()
{
    if (m_hDC)
    {
        CleanUp ();
        ::EndPaint (m_hWnd, &m_PaintStruct);
        m_hDC = NULL;
    }
}

//------------------------------------------------------------------------------//
//------------------------------------------------------------------------------//
CAGClientDC::CAGClientDC (HWND hWnd)
{
    m_hDC = ::GetDC (hWnd);
    Init ();
    m_hWnd = hWnd;
}

//------------------------------------------------------------------------------//
//------------------------------------------------------------------------------//
CAGClientDC::~CAGClientDC ()
{
    if (m_hDC)
    {
        CleanUp ();
        ::ReleaseDC (m_hWnd, m_hDC);
        m_hDC = NULL;
    }
}

//------------------------------------------------------------------------------//
//------------------------------------------------------------------------------//
CAGIC::CAGIC (const char *pszDriver, const char *pszDevice,
  const char *pszOutput, const DEVMODE *pDevMode)
{
    m_hDC = ::CreateIC (pszDriver, pszDevice, pszOutput, pDevMode);
    Init ();
}

//------------------------------------------------------------------------------//
//------------------------------------------------------------------------------//
CAGDIBSectionDC::CAGDIBSectionDC (const BITMAPINFO *pbmi, UINT iUsage,
  BYTE **ppvBits)
{
    m_hDC = ::CreateCompatibleDC (NULL);
    m_hBitmap = ::CreateDIBSection (m_hDC, pbmi, iUsage, (void **) ppvBits,
      NULL, 0);
    m_hOldBitmap = (HBITMAP) ::SelectObject (m_hDC, m_hBitmap);
    Init ();
}

//------------------------------------------------------------------------------//
//------------------------------------------------------------------------------//
CAGDIBSectionDC::~CAGDIBSectionDC ()
{
    if (m_hDC)
```

```
        else if (bNoXStretch)
            memcpy (pDstBits, pSrcBits, dwDstWidthBytes);
        else
        {
            fixPosX = (fixSrcStepX >> 1);

            if (pbi->bmiHeader.biBitCount == 8)
            {
                for (int x = 0; x < nDstWidth; x++)
                {
                    pDstBits[x] = pSrcBits[FixedToInt (fixPosX)];
                    fixPosX += fixSrcStepX;
                }
            }
            else if (pbi->bmiHeader.biBitCount == 24)
            {
                for (int x = 0; x < nDstWidth; x++)
                {
                    wSrcPosX = (WORD) (FixedToInt (fixPosX) * 3);
                    int xDstPos = x * 3;
                    pDstBits[xDstPos] = pSrcBits[wSrcPosX];
                    pDstBits[xDstPos + 1] = pSrcBits[wSrcPosX + 1];
                    pDstBits[xDstPos + 2] = pSrcBits[wSrcPosX + 2];
                    fixPosX += fixSrcStepX;
                }
            }
            else if (pbi->bmiHeader.biBitCount == 4)
            {
                for (int x = 0; x < nDstWidth; x++)
                {
                    wSrcPosX = FixedToInt (fixPosX);
                    if (wSrcPosX % 2)
                        pDstBits[x] = (BYTE) (pSrcBits[wSrcPosX >> 1] & 0x0f);
                    else
                        pDstBits[x] = (BYTE) (pSrcBits[wSrcPosX >> 1] >> 4);
                    fixPosX += fixSrcStepX;
                }
            }
            else if (pbi->bmiHeader.biBitCount == 1)
            {
                for (int x = 0; x < nDstWidth; x++)
                {
                    wSrcPosX = FixedToInt (fixPosX);
                    int nBitPos = wSrcPosX % 8;
                    pDstBits[x] = (BYTE) ((pSrcBits[wSrcPosX / 8] >> (7 - nBitPos)) & 0x01);
                    fixPosX += fixSrcStepX;
                }
            }
        }

        wPrevPosY = wSrcPosY;
        pPrevDstBits = pDstBits;

        if (++nLinesToBlt >= nMaxLinesPerBlt)
        {
            ::StretchDIBits (m_hDC, nDstX, nDstY, nDstWidth, nLinesToBlt,
              0, 0, nDstWidth, nLinesToBlt, m_pBltBuf, (BITMAPINFO *) &NewHdr.bih,
              iUsage, SRCCOPY);
            nDstY += nLinesToBlt;
            nLinesToBlt = 0;
            pDstBits = m_pBltBuf + ((nMaxLinesPerBlt - 1) * dwDstWidthBytes);
        }
        else
            pDstBits -= dwDstWidthBytes;

        fixPosY += fixSrcStepY;
    }

    if (nLinesToBlt)
    {
        NewHdr.bih.biHeight = nLinesToBlt;
        pDstBits = m_pBltBuf + ((nMaxLinesPerBlt - nLinesToBlt) * dwDstWidthBytes);
        ::StretchDIBits (m_hDC, nDstX, nDstY, nDstWidth, nLinesToBlt,
    {
        CleanUp ();
        ::SelectObject (m_hDC, m_hOldBitmap);
        ::DeleteObject (m_hBitmap);
        ::DeleteDC (m_hDC);
        m_hDC = NULL;
    }
}
```

```
// Ctp.h : Declaration of the CCtp

#ifndef __CTP_H_
#define __CTP_H_

#include "CtlPanel.h"
#include "AGDoc.h"
#include "AGSym.h"
#include "AGDC.h"
#include "Font.h"

#include "npapi.h"

class CCtp : public CWindowImpl<CCtp>
{
public:
    CCtp()
    {
        m_pCtlPanel = NULL;
        m_pClientDC = NULL;
        m_pAGDoc = NULL;
        m_pText = NULL;
        SetRect (&m_PageRect, 0, 0, 0, 0);
        SetRect (&m_ShadowRect, 0, 0, 0, 0);
        m_pDownloadData = NULL;
        m_dwDownloadSize = 0;
        m_hBitmap = NULL;
        m_szFontURL[0] = 0;
        m_pNPPInstance = NULL;
        m_bHasFocus = false;
    }

BEGIN_MSG_MAP(CCtp)
    MESSAGE_HANDLER(WM_CREATE, OnCreate)
    MESSAGE_HANDLER(WM_DESTROY, OnDestroy)
    MESSAGE_HANDLER(WM_ERASEBKGND, OnEraseBkgnd)
    MESSAGE_HANDLER(WM_PAINT, OnPaint)
    MESSAGE_HANDLER(WM_CHAR, OnChar)
    MESSAGE_HANDLER(WM_KEYDOWN, OnKeyDown)
    MESSAGE_HANDLER(WM_KEYUP, OnKeyUp)
    MESSAGE_HANDLER(WM_LBUTTONDBLCLK, OnLButtonDblClk)
    MESSAGE_HANDLER(WM_LBUTTONDOWN, OnLButtonDown)
    MESSAGE_HANDLER(WM_LBUTTONUP, OnLButtonUp)
    MESSAGE_HANDLER(WM_MOUSEMOVE, OnMouseMove)
    MESSAGE_HANDLER(WM_KILLFOCUS, OnKillFocus)
    MESSAGE_HANDLER(WM_SETFOCUS, OnSetFocus)
    MESSAGE_HANDLER(WM_TIMER, OnTimer)
END_MSG_MAP()


public:


    LRESULT OnChar (UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM /*lParam*/, BOOL & /*bHandled*/);
    LRESULT OnCreate(UINT, WPARAM, LPARAM, BOOL&);
    LRESULT OnDestroy(UINT, WPARAM, LPARAM, BOOL&);
    HRESULT OnDraw(ATL_DRAWINFO& di);
    LRESULT OnEraseBkgnd(UINT, WPARAM, LPARAM, BOOL &bHandled)
    {
        bHandled = TRUE;
        return (TRUE);
    }

    LRESULT OnKeyDown (UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM /*lParam*/, BOOL & /*bHandled*/);
    LRESULT OnKeyUp (UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM /*lParam*/, BOOL & /*bHandled*/);
    LRESULT OnLButtonDblClk (UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM /*lParam*/, BOOL & /*bHandled*
*/);
    LRESULT OnLButtonDown (UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM /*lParam*/, BOOL & /*bHandled*/)
;
    LRESULT OnLButtonUp (UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM /*lParam*/, BOOL & /*bHandled*/);
    LRESULT OnMouseMove (UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM /*lParam*/, BOOL & /*bHandled*/);
    LRESULT OnKillFocus (UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM /*lParam*/, BOOL & /*bHandled*/);
    LRESULT OnSetFocus (UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM /*lParam*/, BOOL & /*bHandled*/);
    LRESULT OnTimer (UINT /*uMsg*/, WPARAM /*wParam*/, LPARAM /*lParam*/, BOOL & /*bHandled*/);

    LRESULT OnPaint (UINT, WPARAM, LPARAM, BOOL &bHandled)
    {
        ATL_DRAWINFO di;
        PAINTSTRUCT ps;
        BeginPaint(&ps);
        di.hdcDraw = ps.hdc;
        RECT ClientRect;
        GetClientRect (&ClientRect);
        di.prcBounds = (const RECTL *) &ClientRect;
        SaveDC(di.hdcDraw);
        OnDraw(di);
        RestoreDC(di.hdcDraw, -1);
        EndPaint(&ps);

        bHandled = TRUE;
        return (TRUE);
    }

    void CreateBackPage ();
    void DrawEditRect (CAGDC *pDC);
    void FileData (BYTE *pBytes, DWORD dwLen);
    void FileEnd ();
    void FileStart ();
    void FontData (const char *pszFontFile, BYTE *pBytes, DWORD dwLen);
    void FontEnd (const char *pszFontFile);
    void FontStart (const char *pszFontFile);
    BOOL GetAmbientDisplayName (BSTR &)
        { return (FALSE); }
    void GetAmbientUserMode (BOOL &bUserMode)
        { bUserMode = TRUE; }

    CFontList &GetFontList ()
        { return (m_FontList); }
    CAGSymImage *GetImage (int nID);
    CAGText *GetText ()
        { return (m_pText); }
    bool HasFocus ()
        { return (m_bHasFocus); }
    void NewPage();
    void SetFontURL (const char *pszFontURL)
        { lstrcpy (m_szFontURL, pszFontURL); }
    void SetNPPInstance (NPP pInstance)
        { m_pNPPInstance = pInstance; }
    void StartDownloadFont (const char *pszFontName)
    {
        char szFontURL[_MAX_PATH];
        lstrcpy (szFontURL, m_szFontURL);
        lstrcat (szFontURL, pszFontName);

        NPN_GetURL (m_pNPPInstance, szFontURL, NULL);
    }
    void StartEdit (CAGSymText *pText, POINT Pt, bool bClick);
    void StopEdit ();

protected:
    CCtlPanel            *m_pCtlPanel;
    CAGClientDC          *m_pClientDC;
    CAGDoc               *m_pAGDoc;
    CAGSymText           *m_pText;
    RECT                 m_PageRect;
    RECT                 m_ShadowRect;
    CAGMatrix            m_ViewMatrix;
    BOOL                 m_bWindowOnly;
    BYTE                 *m_pDownloadData;
    DWORD                m_dwDownloadSize;
    HBITMAP              m_hBitmap;
    char                 m_szFontURL[_MAX_PATH];
    NPP                  m_pNPPInstance;
    CFontList            m_FontList;
    FONTDOWNLOADARRAY    m_FontDownloadArray;
    bool                    m_bHasFocus;
};


#endif //__CTP_H_
```

```
# Microsoft Developer Studio Project File - Name="NPCTP" - Package Owner=<4>
# Microsoft Developer Studio Generated Build File, Format Version 5.00
# ** DO NOT EDIT **

# TARGTYPE "Win32 (x86) Dynamic-Link Library" 0x0102

CFG=NPCTP - Win32 Debug
!MESSAGE This is not a valid makefile. To build this project using NMAKE,
!MESSAGE use the Export Makefile command and run
!MESSAGE
!MESSAGE NMAKE /f "NPCTP.MAK".
!MESSAGE
!MESSAGE You can specify a configuration when running NMAKE
!MESSAGE by defining the macro CFG on the command line. For example:
!MESSAGE
!MESSAGE NMAKE /f "NPCTP.MAK" CFG="NPCTP - Win32 Debug"
!MESSAGE
!MESSAGE Possible choices for configuration are:
!MESSAGE
!MESSAGE "NPCTP - Win32 Release" (based on "Win32 (x86) Dynamic-Link Library")
!MESSAGE "NPCTP - Win32 Debug" (based on "Win32 (x86) Dynamic-Link Library")
!MESSAGE

# Begin Project
# PROP Scc_ProjName ""$/NPCTP", BAAAAAAA"
# PROP Scc_LocalPath "."
CPP=cl.exe
MTL=midl.exe
RSC=rc.exe

!IF  "$(CFG)" == "NPCTP - Win32 Release"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "Release"
# PROP BASE Intermediate_Dir "Release"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Release"
# PROP Intermediate_Dir "Release"
# PROP Ignore_Export_Lib 1
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MT /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_WINDOWS" /YX /FD /c
# ADD CPP /nologo /Zp2 /MT /W3 /GX /O1 /I "\npsdk\include" /I "..\ZLib" /I "..\Stonehnd" /D "WIN32"
/D "NDEBUG" /D "_WINDOWS" /Yu"stdafx.h" /FD /c
# ADD BASE MTL /nologo /D "NDEBUG" /mktyplib203 /o NUL /win32
# ADD MTL /nologo /D "NDEBUG" /mktyplib203 /o NUL /win32
# ADD BASE RSC /l 0x409 /d "NDEBUG"
# ADD RSC /l 0x409 /d "NDEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.lib shell32.l
ib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo /subsystem:windows /dll /machine:
I386
# ADD LINK32 kernel32.lib user32.lib gdi32.lib comdlg32.lib comctl32.lib winspool.lib /nologo /subsy
stem:windows /dll /pdb:none /machine:I386

!ELSEIF  "$(CFG)" == "NPCTP - Win32 Debug"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "Debug"
# PROP BASE Intermediate_Dir "Debug"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Debug"
# PROP Intermediate_Dir "Debug"
# PROP Ignore_Export_Lib 1
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MTd /W3 /Gm /GX /Zi /Od /D "WIN32" /D "_DEBUG" /D "_WINDOWS" /YX /FD /c
# ADD CPP /nologo /Zp2 /MTd /W3 /Gm /GX /Zi /Od /I "\npsdk\include" /I "..\ZLib" /I "..\Stonehnd" /D
 "WIN32" /D "_DEBUG" /D "_WINDOWS" /FR /Yu"stdafx.h" /FD /c
# ADD BASE MTL /nologo /D "_DEBUG" /mktyplib203 /o NUL /win32
# ADD MTL /nologo /D "_DEBUG" /mktyplib203 /o NUL /win32
# ADD BASE RSC /l 0x409 /d "_DEBUG"
# ADD RSC /l 0x409 /d "_DEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.lib shell32.l
ib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo /subsystem:windows /dll /debug /m
achine:I386 /pdbtype:sept
# ADD LINK32 kernel32.lib user32.lib gdi32.lib comdlg32.lib comctl32.lib winspool.lib /nologo /subsy
stem:windows /dll /debug /machine:I386 /pdbtype:sept

!ENDIF

# Begin Target

# Name "NPCTP - Win32 Release"
# Name "NPCTP - Win32 Debug"
# Begin Group "Source Files"

# PROP Default_Filter "cpp;rc;def"
# Begin Source File

SOURCE=.\AGDC.cpp
# End Source File
# Begin Source File

SOURCE=.\AGDoc.cpp
# End Source File
# Begin Source File

SOURCE=.\AGLayer.cpp
# End Source File
# Begin Source File

SOURCE=.\AGMatrix.cpp
# End Source File
# Begin Source File

SOURCE=.\AGPage.cpp
# End Source File
# Begin Source File

SOURCE=.\AGSym.cpp
# End Source File
# Begin Source File

SOURCE=.\AGText.cpp
# End Source File
# Begin Source File

SOURCE=.\CtlPanel.cpp
# End Source File
# Begin Source File

SOURCE=.\Ctp.cpp
# End Source File
# Begin Source File

SOURCE=.\dblside.cpp
# End Source File
# Begin Source File

SOURCE=.\Font.cpp
# End Source File
# Begin Source File

SOURCE=.\npctp.def
```

```
# End Source File
# Begin Source File

SOURCE=.\NPCTP.rc

!IF  "$(CFG)" == "NPCTP - Win32 Release"

!ELSEIF  "$(CFG)" == "NPCTP - Win32 Debug"

!ENDIF

# End Source File
# Begin Source File

SOURCE=.\Npshell.cpp
# End Source File
# Begin Source File

SOURCE=.\npwin.cpp
# End Source File
# Begin Source File

SOURCE=.\StdAfx.cpp
# ADD CPP /Yc"stdafx.h"
# End Source File
# Begin Source File

SOURCE=.\WaitDlg.cpp
# End Source File
# End Group
# Begin Group "Header Files"

# PROP Default_Filter "h"
# Begin Source File

SOURCE=.\AGDC.h
# End Source File
# Begin Source File

SOURCE=.\AGDib.h
# End Source File
# Begin Source File

SOURCE=.\AGDoc.h
# End Source File
# Begin Source File

SOURCE=.\AGLayer.h
# End Source File
# Begin Source File

SOURCE=.\AGMatrix.h
# End Source File
# Begin Source File

SOURCE=.\AGPage.h
# End Source File
# Begin Source File

SOURCE=.\AGSym.h
# End Source File
# Begin Source File

SOURCE=.\AGText.h
# End Source File
# Begin Source File

SOURCE=.\CtlPanel.h
# End Source File
# Begin Source File

SOURCE=.\Ctp.h
# End Source File
# Begin Source File

SOURCE=.\dblside.h
# End Source File
# Begin Source File

SOURCE=.\Font.h
# End Source File
# Begin Source File

SOURCE=.\propsht.h
# End Source File
# Begin Source File

SOURCE=.\resource.h
# End Source File
# Begin Source File

SOURCE=.\StdAfx.h
# End Source File
# Begin Source File

SOURCE=.\version.h
# End Source File
# Begin Source File

SOURCE=.\WaitDlg.h
# End Source File
# End Group
# Begin Group "Resource Files"

# PROP Default_Filter ""
# Begin Source File

SOURCE=.\Res\1up.bmp
# End Source File
# Begin Source File

SOURCE=.\Res\1up2down.bmp
# End Source File
# Begin Source File

SOURCE=.\Res\2down.bmp
# End Source File
# Begin Source File

SOURCE=.\Res\2up.bmp
# End Source File
# Begin Source File

SOURCE=.\Res\3up.bmp
# End Source File
# Begin Source File

SOURCE=.\Res\AGLogo.agi
# End Source File
# Begin Source File

SOURCE=".\Res\C&PLogo.agi"
# End Source File
# Begin Source File

SOURCE=.\Res\Cacfc___.ttz
# End Source File
# Begin Source File

SOURCE=.\Res\Version.rc2
# End Source File
# End Group
# End Target
# End Project
```

```
LIBRARY   NPCTP

EXPORTS
  NP_GetEntryPoints   @1
  NP_Initialize       @2
  NP_Shutdown         @3
```

09659370.073400

```
# Microsoft Developer Studio Project File - Name="NPCTP" - Package
 Owner=<4>
# Microsoft Developer Studio Generated Build File, Format Version
6.00
# ** DO NOT EDIT **

# TARGTYPE "Win32 (x86) Dynamic-Link Library" 0x0102

CFG=NPCTP - Win32 Debug
!MESSAGE This is not a valid makefile. To build this project using
 NMAKE,
!MESSAGE use the Export Makefile command and run
!MESSAGE
!MESSAGE NMAKE /f "NPCTP.MAK".
!MESSAGE
!MESSAGE You can specify a configuration when running NMAKE
!MESSAGE by defining the macro CFG on the command line. For exampl
e:
!MESSAGE
!MESSAGE NMAKE /f "NPCTP.MAK" CFG="NPCTP - Win32 Debug"
!MESSAGE
!MESSAGE Possible choices for configuration are:
!MESSAGE
!MESSAGE "NPCTP - Win32 Release" (based on "Win32 (x86) Dynamic-Li
nk Library")
!MESSAGE "NPCTP - Win32 Debug" (based on "Win32 (x86) Dynamic-Link
 Library")
!MESSAGE

# Begin Project
# PROP AllowPerConfigDependencies 0
# PROP Scc_ProjName ""$/NPCTP", BAAAAAAA"
# PROP Scc_LocalPath "."
CPP=cl.exe
MTL=midl.exe
RSC=rc.exe

!IF  "$(CFG)" == "NPCTP - Win32 Release"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "Release"
# PROP BASE Intermediate_Dir "Release"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Release"
# PROP Intermediate_Dir "Release"
# PROP Ignore_Export_Lib 1
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MT /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "
_WINDOWS" /YX /FD /c
# ADD CPP /nologo /Zp2 /MT /W3 /GX /O1 /I "\npsdk\include" /I "..\
ZLib" /I "..\Stonehnd" /D "WIN32" /D "NDEBUG" /D "_WINDOWS" /Yu"st
dafx.h" /FD /c
# ADD BASE MTL /nologo /D "NDEBUG" /mktyplib203 /o "NUL" /win32
# ADD MTL /nologo /D "NDEBUG" /mktyplib203 /o "NUL" /win32
# ADD BASE RSC /l 0x409 /d "NDEBUG"
# ADD RSC /l 0x409 /d "NDEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib c
omdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.l
ib odbc32.lib odbccp32.lib /nologo /subsystem:windows /dll /machin
e:I386
# ADD LINK32 kernel32.lib user32.lib gdi32.lib comdlg32.lib comctl
32.lib winspool.lib /nologo /subsystem:windows /dll /pdb:none /mac
hine:I386

!ELSEIF  "$(CFG)" == "NPCTP - Win32 Debug"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "Debug"
# PROP BASE Intermediate_Dir "Debug"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Debug"
# PROP Intermediate_Dir "Debug"
# PROP Ignore_Export_Lib 1
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MTd /W3 /Gm /GX /Zi /Od /D "WIN32" /D "_DE
BUG" /D "_WINDOWS" /YX /FD /c
# ADD CPP /nologo /Zp2 /MTd /W3 /Gm /GX /ZI /Od /I "\npsdk\include
" /I "..\ZLib" /I "..\Stonehnd" /D "WIN32" /D "_DEBUG" /D "_WINDOW
S" /FR /Yu"stdafx.h" /FD /c
# ADD BASE MTL /nologo /D "_DEBUG" /mktyplib203 /o "NUL" /win32
# ADD MTL /nologo /D "_DEBUG" /mktyplib203 /o "NUL" /win32
# ADD BASE RSC /l 0x409 /d "_DEBUG"
# ADD RSC /l 0x409 /d "_DEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib c
omdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.l
ib odbc32.lib odbccp32.lib /nologo /subsystem:windows /dll /debug
/machine:I386 /pdbtype:sept
# ADD LINK32 kernel32.lib user32.lib gdi32.lib comdlg32.lib comctl
32.lib winspool.lib /nologo /subsystem:windows /dll /debug /machin
e:I386 /pdbtype:sept

!ENDIF

# Begin Target

# Name "NPCTP - Win32 Release"
# Name "NPCTP - Win32 Debug"
# Begin Group "Source Files"

# PROP Default_Filter "cpp;rc;def"
# Begin Source File

SOURCE=.\AGDC.cpp
# End Source File
# Begin Source File

SOURCE=.\AGDoc.cpp
# End Source File
# Begin Source File

SOURCE=.\AGLayer.cpp
# End Source File
# Begin Source File

SOURCE=.\AGMatrix.cpp
# End Source File
# Begin Source File

SOURCE=.\AGPage.cpp
# End Source File
# Begin Source File

SOURCE=.\AGSym.cpp
# End Source File
# Begin Source File

SOURCE=.\AGText.cpp
# End Source File
# Begin Source File

SOURCE=.\CtlPanel.cpp
# End Source File
# Begin Source File

SOURCE=.\Ctp.cpp
# End Source File
# Begin Source File

SOURCE=.\dblside.cpp
# End Source File
# Begin Source File

SOURCE=.\Font.cpp
# End Source File
# Begin Source File

SOURCE=.\npctp.def
# End Source File
# Begin Source File

SOURCE=.\NPCTP.rc
# End Source File
# Begin Source File

SOURCE=.\Npshell.cpp
# End Source File
# Begin Source File

SOURCE=.\npwin.cpp
# End Source File
# Begin Source File

SOURCE=.\StdAfx.cpp
# ADD CPP /Yc"stdafx.h"
# End Source File
# Begin Source File

SOURCE=.\WaitDlg.cpp
# End Source File
# End Group
# Begin Group "Header Files"

# PROP Default_Filter "h"
# Begin Source File

SOURCE=.\AGDC.h
# End Source File
# Begin Source File

SOURCE=.\AGDib.h
# End Source File
# Begin Source File

SOURCE=.\AGDoc.h
# End Source File
# Begin Source File

SOURCE=.\AGLayer.h
# End Source File
# Begin Source File

SOURCE=.\AGMatrix.h
# End Source File
# Begin Source File

SOURCE=.\AGPage.h
# End Source File
# Begin Source File

SOURCE=.\AGSym.h
# End Source File
# Begin Source File

SOURCE=.\AGText.h
# End Source File
# Begin Source File

SOURCE=.\CtlPanel.h
# End Source File
# Begin Source File

SOURCE=.\Ctp.h
# End Source File
# Begin Source File

SOURCE=.\dblside.h
```

```
# End Source File
# Begin Source File

SOURCE=.\Font.h
# End Source File
# Begin Source File

SOURCE=.\propsht.h
# End Source File
# Begin Source File

SOURCE=.\resource.h
# End Source File
# Begin Source File

SOURCE=.\StdAfx.h
# End Source File
# Begin Source File

SOURCE=.\version.h
# End Source File
# Begin Source File

SOURCE=.\WaitDlg.h
# End Source File
# End Group
# Begin Group "Resource Files"

# PROP Default_Filter ""
# Begin Source File

SOURCE=.\Res\1up.bmp
# End Source File
# Begin Source File

SOURCE=.\Res\1up2down.bmp
# End Source File
# Begin Source File

SOURCE=.\Res\2down.bmp
# End Source File
# Begin Source File

SOURCE=.\Res\2up.bmp
# End Source File
# Begin Source File
```

```
SOURCE=.\Res\3up.bmp
# End Source File
# Begin Source File

SOURCE=.\Res\AGLogo.agi
# End Source File
# Begin Source File

SOURCE=".\Res\C&PLogo.agi"
# End Source File
# Begin Source File

SOURCE=.\Res\Cacfc___.ttz
# End Source File
# Begin Source File

SOURCE=.\Res\Version.rc2
# End Source File
# End Group
# End Target
# End Project
```

```
Microsoft Developer Studio Workspace File, Format Version 6.00
# WARNING: DO NOT EDIT OR DELETE THIS WORKSPACE FILE!

#####################################################################
#############

Project: "NPCTP"=.\NPCTP.DSP - Package Owner=<4>

Package=<5>
{{{
    begin source code control
    "$/NPCTP", TAAAAAAA
    .
    end source code control
}}}

Package=<4>
{{{
    Begin Project Dependency
    Project_Dep_Name ZLib
    End Project Dependency
    Begin Project Dependency
    Project_Dep_Name Stonehnd
    End Project Dependency
}}}

#####################################################################
#############

Project: "Stonehnd"=..\Stonehnd\Stonehnd.dsp - Package Owner=<4>

Package=<5>
{{{
    begin source code control
    "$/Stonehnd", CGAAAAAA
    ..\stonehnd
    end source code control
}}}

Package=<4>
{{{
}}}

#####################################################################
#############
```

```
Project: "ZLib"=..\ZLib\ZLib.dsp - Package Owner=<4>

Package=<5>
{{{
    begin source code control
    "$/ZLib", LBAAAAAA
    ..\zlib
    end source code control
}}}

Package=<4>
{{{
}}}

###################################################################
#############

Global:

Package=<5>
{{{
    begin source code control
    "$/NPCTP", TAAAAAAA
    .
    end source code control
}}}

Package=<3>
{{{
}}}

###################################################################
#############
```

```
<html>
<body>
<pre>
<h1>Build Log</h1>
<h3>
--------------------Configuration: NPCTP - Win32 Release--------------------
</h3>
<h3>Command Lines</h3>
Creating command line "rc.exe /l 0x409 /fo"Release/NPCTP.res" /d "NDEBUG" "C:\Work\CrtPrt\NpCtp\NPCT
P.rc""
Creating temporary file "c:\windows\TEMP\RSP6194.TMP" with contents
[
/nologo /Zp2 /MT /W3 /GX /O1 /I "\npsdk\include" /I "..\ZLib" /I "..\Stonehnd" /D "WIN32" /D "NDEBUG
" /D "_WINDOWS" /Fp"Release/NPCTP.pch" /Yu"stdafx.h" /Fo"Release/" /Fd"Release/" /FD /c
"C:\Work\CrtPrt\NpCtp\AGDC.cpp"
"C:\Work\CrtPrt\NpCtp\AGDoc.cpp"
"C:\Work\CrtPrt\NpCtp\AGLayer.cpp"
"C:\Work\CrtPrt\NpCtp\AGMatrix.cpp"
"C:\Work\CrtPrt\NpCtp\AGPage.cpp"
"C:\Work\CrtPrt\NpCtp\AGSym.cpp"
"C:\Work\CrtPrt\NpCtp\AGText.cpp"
"C:\Work\CrtPrt\NpCtp\CtlPanel.cpp"
"C:\Work\CrtPrt\NpCtp\Ctp.cpp"
"C:\Work\CrtPrt\NpCtp\dblside.cpp"
"C:\Work\CrtPrt\NpCtp\Font.cpp"
"C:\Work\CrtPrt\NpCtp\Npshell.cpp"
"C:\Work\CrtPrt\NpCtp\npwin.cpp"
"C:\Work\CrtPrt\NpCtp\WaitDlg.cpp"
]
Creating command line "cl.exe @c:\windows\TEMP\RSP6194.TMP"
Creating temporary file "c:\windows\TEMP\RSP6195.TMP" with contents
[
/nologo /Zp2 /MT /W3 /GX /O1 /I "\npsdk\include" /I "..\ZLib" /I "..\Stonehnd" /D "WIN32" /D "NDEBUG
" /D "_WINDOWS" /Fp"Release/NPCTP.pch" /Yc"stdafx.h" /Fo"Release/" /Fd"Release/" /FD /c
"C:\Work\CrtPrt\NpCtp\StdAfx.cpp"
]
Creating command line "cl.exe @c:\windows\TEMP\RSP6195.TMP"
Creating temporary file "c:\windows\TEMP\RSP6196.TMP" with contents
[
kernel32.lib user32.lib gdi32.lib comdlg32.lib comctl32.lib winspool.lib /nologo /subsystem:windows
/dll /pdb:none /machine:I386 /def:".\npctp.def" /out:"Release/NPCTP.dll" /implib:"Release/NPCTP.lib"

.\Release\AGDC.obj
.\Release\AGDoc.obj
.\Release\AGLayer.obj
.\Release\AGMatrix.obj
.\Release\AGPage.obj
.\Release\AGSym.obj
.\Release\AGText.obj
.\Release\CtlPanel.obj
.\Release\Ctp.obj
.\Release\dblside.obj
.\Release\Font.obj
.\Release\Npshell.obj
.\Release\npwin.obj
.\Release\StdAfx.obj
.\Release\WaitDlg.obj
.\Release\NPCTP.res
\Work\CrtPrt\ZLib\Release\ZLib.lib
\Work\CrtPrt\Stonehnd\Release\Stonehnd.lib
]
Creating command line "link.exe @c:\windows\TEMP\RSP6196.TMP"
<h3>Output Window</h3>
Compiling resources...
Compiling...
StdAfx.cpp
Compiling...
AGDC.cpp
AGDoc.cpp
AGLayer.cpp
AGMatrix.cpp
AGPage.cpp
AGSym.cpp
```

```
AGText.cpp
CtlPanel.cpp
Ctp.cpp
dblside.cpp
Font.cpp
Npshell.cpp
npwin.cpp
WaitDlg.cpp
Generating Code...
C:\Work\CrtPrt\NpCtp\Ctp.cpp(667) : warning C4700: local variable 'bstr' used without having been in
itialized
Linking...
   Creating library Release/NPCTP.lib and object Release/NPCTP.exp
LINK : warning LNK4089: all references to "OLEAUT32.dll" discarded by /OPT:REF



<h3>Results</h3>
NPCTP.dll - 0 error(s), 2 warning(s)
</pre>
</body>
</html>
```

```
//Microsoft Developer Studio generated resource script.
//
#include "resource.h"

#define APSTUDIO_READONLY_SYMBOLS
/////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"

/////////////////////////////////////////////////////////////////////////////
#undef APSTUDIO_READONLY_SYMBOLS

/////////////////////////////////////////////////////////////////////////////
// English (U.S.) resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
#pragma code_page(1252)
#endif //_WIN32

#ifdef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
//
// TEXTINCLUDE
//

1 TEXTINCLUDE DISCARDABLE
BEGIN
    "resource.h\0"
END

2 TEXTINCLUDE DISCARDABLE
BEGIN
    "#include ""afxres.h""\r\n"
END

3 TEXTINCLUDE DISCARDABLE
BEGIN
    "#include ""Res\\Version.rc2""\r\n"
    "\0"
END

#endif    // APSTUDIO_INVOKED


//////////////////////////////////////////////////////////////////
///////////
//
// Dialog
//

IDD_CTLPANEL DIALOGEX 0, 0, 137, 169
STYLE WS_CHILD
EXSTYLE WS_EX_TRANSPARENT
FONT 12, "CAC Futura Casual", 0, 0, 0x1
BEGIN
    LTEXT           "View Card Panel",IDC_STATIC,3,1,122,8
    CONTROL         "Front",IDC_PAGE1,"Button",BS_AUTORADIOBUTTON,
10,8,73,10
    CONTROL         "Inside Left",IDC_PAGE2,"Button",BS_AUTORADIOB
UTTON,10,
                    15,73,10
    CONTROL         "Inside Right",IDC_PAGE3,"Button",BS_AUTORADIO
BUTTON,10,
                    23,73,10
    CONTROL         "Back",IDC_PAGE4,"Button",BS_AUTORADIOBUTTON,1
0,31,73,10
    LTEXT           "Font",IDC_STATIC,3,44,122,8,0,WS_EX_TRANSPARE
NT
    COMBOBOX        IDC_FONT,10,53,122,112,CBS_DROPDOWNLIST |
                    CBS_OWNERDRAWFIXED | CBS_SORT | CBS_HASSTRINGS
 |
                    WS_VSCROLL | WS_GROUP | WS_TABSTOP
    LTEXT           "Point Size",IDC_STATIC,3,68,122,8,0,WS_EX_TRA
NSPARENT
    COMBOBOX        IDC_PTSIZE,10,77,38,88,CBS_DROPDOWNLIST | WS_V
SCROLL |
                    WS_GROUP | WS_TABSTOP
    LTEXT           "Text Color",IDC_STATIC,3,92,122,8,0,WS_EX_TRA
NSPARENT
    COMBOBOX        IDC_COLOR,10,101,67,67,CBS_DROPDOWNLIST |
                    CBS_OWNERDRAWFIXED | WS_VSCROLL | WS_TABSTOP,
                    WS_EX_TRANSPARENT
    LTEXT           "Text Alignment",IDC_STATIC,3,117,122,8,0,
                    WS_EX_TRANSPARENT
    CONTROL              "Left",IDC_LEFT,"Button",BS_AUTORADIOBUTTON |
WS_GROUP |
                         WS_TABSTOP,10,124,48,10,WS_EX_TRANSPARENT
    CONTROL              "Center",IDC_CENTER,"Button",BS_AUTORADIOBUTTO
N,10,132,
                         48,10,WS_EX_TRANSPARENT
    CONTROL              "Right",IDC_RIGHT,"Button",BS_AUTORADIOBUTTON,
10,140,48,
                         10,WS_EX_TRANSPARENT
    PUSHBUTTON           "Print",IDC_PRINT,3,154,50,11
END

1538 DIALOG DISCARDABLE  32, 32, 287, 157
STYLE DS_MODALFRAME | DS_3DLOOK | DS_CONTEXTHELP | WS_POPUP | WS_V
ISIBLE |
    WS_CAPTION | WS_SYSMENU
CAPTION "Print"
FONT 8, "MS Sans Serif"
BEGIN
    GROUPBOX             "Printer",1075,8,4,272,84,WS_GROUP
    LTEXT                "&Name:",1093,16,20,36,8
    COMBOBOX             1139,52,18,152,152,CBS_DROPDOWNLIST | CBS_SORT
 |
                         WS_VSCROLL | WS_GROUP | WS_TABSTOP
    PUSHBUTTON           "&Properties",1025,212,17,60,14,WS_GROUP
    LTEXT                "Status:",1095,16,36,36,10,SS_NOPREFIX
    CONTROL              "",1099,"Static",SS_LEFTNOWORDWRAP | SS_NOPREF
IX |
                         WS_GROUP,52,36,224,10
    LTEXT                "Type:",1094,16,48,36,10,SS_NOPREFIX
    CONTROL              "",1098,"Static",SS_LEFTNOWORDWRAP | SS_NOPREF
IX |
                         WS_GROUP,52,48,224,10
    LTEXT                "Where:",1097,16,60,36,10,SS_NOPREFIX
    CONTROL              "",1101,"Static",SS_LEFTNOWORDWRAP | SS_NOPREF
IX |
                         WS_GROUP,52,60,224,10
    LTEXT                "Comment:",1096,16,72,36,10,SS_NOPREFIX
    CONTROL              "",1100,"Static",SS_LEFTNOWORDWRAP | SS_NOPREF
IX |
                         WS_GROUP,52,72,152,10
    CONTROL              "Print to fi&le",1040,"Button",BS_AUTOCHECKBOX
 |
                         WS_GROUP | WS_TABSTOP,212,70,64,12
    GROUPBOX             "Print Format",IDC_STATIC,8,93,136,39,WS_GROUP
    CONTROL              "Single-fold",IDC_SINGLEFOLD,"Button",BS_AUTOR
```

```
ADIOBUTTON,
                         15,104,80,10
    CONTROL              "Quarter-fold",IDC_QUARTERFOLD,"Button",
                         BS_AUTORADIOBUTTON,15,116,80,10
    LTEXT                "Number of &copies:",1092,162,105,68,8
    EDITTEXT             1154,234,103,32,12,ES_NUMBER | WS_GROUP
    DEFPUSHBUTTON        "OK",IDOK,180,137,48,14,WS_GROUP
    PUSHBUTTON           "Cancel",IDCANCEL,232,137,48,14
    GROUPBOX             "Copies",IDC_STATIC,152,93,128,39,WS_GROUP
    CONTROL              "Run Double-Sided Printing Test",IDC_DBLSIDE,"
Button",
                         BS_AUTOCHECKBOX | WS_TABSTOP,8,140,115,10
END

IDD_WAITDLG DIALOG DISCARDABLE  0, 0, 186, 44
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION
CAPTION "Printing"
FONT 8, "MS Sans Serif"
BEGIN
    CTEXT                "Your document is being printed.  Please wait.
..",
                         IDC_STATIC,6,14,173,8
END

IDD_DBLSIDEINTRO DIALOG DISCARDABLE  0, 0, 346, 105
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Double-Sided Printing Test"
FONT 8, "MS Sans Serif"
BEGIN
    LTEXT                "To help guide you through printing a single-f
old card on both sides of a page, some information needs to be gat
hered about the way paper feeds through your printer.",
                         IDC_STATIC,0,4,345,24
    LTEXT                "This print test will use one piece of paper.
 It will only need to be run once for a particular printer.",
                         IDC_STATIC,0,32,345,24
    LTEXT                "Click Next when you are ready to print the te
st page.",
                         IDC_STATIC,0,97,345,8
END

IDD_DBLSIDESTEP1 DIALOG DISCARDABLE  0, 0, 346, 105
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Double-Sided Printing Test"
FONT 8, "MS Sans Serif"
BEGIN
```

```
    LTEXT           "After the test page has printed, it must be p
rinted on once more to complete this test.",
                    IDC_STATIC,0,4,345,8
    LTEXT           "Please put the page back into the printer wit
h the printed side UP and the arrow pointing TOWARD the printer.",
                    IDC_STATIC,0,24,345,24
    LTEXT           "Click Next when you are ready to print.",IDC_
STATIC,0,
                    97,345,8
END

IDD_DBLSIDESTEP2 DIALOG DISCARDABLE  0, 0, 346, 105
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Double-Sided Printing Test"
FONT 8, "MS Sans Serif"
BEGIN
    LTEXT           "Please click on the option below that matches
 your printed page.",
                    IDC_STATIC,0,4,345,8
    CONTROL         "",IDC_FRAME1,"Static",SS_ETCHEDFRAME,0,20,56,
68
    CONTROL         "",IDC_FRAME2,"Static",SS_ETCHEDFRAME,66,20,56
,68
    CONTROL         "",IDC_FRAME3,"Static",SS_ETCHEDFRAME,132,20,1
02,68
    CONTROL         "",IDC_FRAME4,"Static",SS_ETCHEDFRAME,244,20,1
02,68
    CONTROL         210,IDC_STATIC,"Static",SS_BITMAP,8,26,40,49
    CONTROL         212,IDC_STATIC,"Static",SS_BITMAP,74,26,40,49
    CONTROL         208,IDC_STATIC,"Static",SS_BITMAP,140,26,40,49
    CONTROL         208,IDC_STATIC,"Static",SS_BITMAP,252,26,40,49
    CONTROL         209,IDC_STATIC,"Static",SS_BITMAP,186,26,40,49
    CONTROL         211,IDC_STATIC,"Static",SS_BITMAP,298,26,40,49
    LTEXT           "Front",IDC_STATIC,152,76,17,8
    LTEXT           "Back",IDC_STATIC,197,76,18,8
    LTEXT           "Front",IDC_STATIC,264,77,17,8
    LTEXT           "Back",IDC_STATIC,309,77,18,8
    LTEXT           "Click Next to continue.",IDC_STATIC,0,97,345,
8
END

IDD_DBLSIDEEND DIALOG DISCARDABLE  0, 0, 346, 105
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Double-Sided Printing Test"
FONT 8, "MS Sans Serif"
BEGIN
```

```
    LTEXT           "The double-sided printing test is complete.",
IDC_STATIC,
                    0,4,345,8
    LTEXT           "It is now time to print your card.",IDC_STATI
C,0,24,345,
                    8
    LTEXT           "Click Finish when you are ready.",IDC_STATIC,
0,97,345,8
END


////////////////////////////////////////////////////////////////////
///////////
//
// DESIGNINFO
//

#ifdef APSTUDIO_INVOKED
GUIDELINES DESIGNINFO DISCARDABLE
BEGIN
    IDD_WAITDLG, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 179
        TOPMARGIN, 7
        BOTTOMMARGIN, 37
    END
END
#endif    // APSTUDIO_INVOKED


////////////////////////////////////////////////////////////////////
///////////
//
// AGIMAGE
//

IDR_AGLOGO              AGIMAGE DISCARDABLE     "Res\\AGLogo.agi"
IDR_CPLOGO              AGIMAGE DISCARDABLE     "Res\\C&PLogo.agi"

////////////////////////////////////////////////////////////////////
///////////
//
// TTZ
//
```

```
IDR_CACFC                  TTZ     DISCARDABLE      "Res\\Cacfc___.ttz
"

//////////////////////////////////////////////////////////////////
///////////
//
// Bitmap
//

IDB_1UP                    BITMAP  DISCARDABLE      "Res\\1up.bmp"
IDB_2UP                    BITMAP  DISCARDABLE      "Res\\2up.bmp"
IDB_3UP                    BITMAP  DISCARDABLE      "Res\\3up.bmp"
IDB_2DOWN                  BITMAP  DISCARDABLE      "Res\\2down.bmp"
IDB_1UP2DOWN               BITMAP  DISCARDABLE      "Res\\1up2down.bmp
"
#endif    // English (U.S.) resources
//////////////////////////////////////////////////////////////////
///////////


#ifndef APSTUDIO_INVOKED
//////////////////////////////////////////////////////////////////
///////////
//
// Generated from the TEXTINCLUDE 3 resource.
//
#include "Res\Version.rc2"

//////////////////////////////////////////////////////////////////
///////////
#endif    // not APSTUDIO_INVOKED
```

```
//
// npshell.cpp - Plug-in methods called from Netscape.
//

#include "stdafx.h"
#include <string.h>
#include "npapi.h"
#include "Ctp.h"

#include "scappint.h"

CComModule _Module;

BEGIN_OBJECT_MAP(ObjectMap)
END_OBJECT_MAP()


BOOL WINAPI DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID /*lpvReserved*/)
{
    switch (fdwReason)
    {
        case DLL_PROCESS_ATTACH:
            _Module.Init(ObjectMap, hinstDLL);
            break;

        case DLL_PROCESS_DETACH:
            _Module.Term();
            break;

        case DLL_THREAD_ATTACH:
        case DLL_THREAD_DETACH:
            break;
    }
    return TRUE;
}


//
// NPP_Initialize
//
NPError NPP_Initialize(void)
{
    SCENG_Init();
    return NPERR_NO_ERROR;
}

//
// NPP_Shutdown
//
void NPP_Shutdown(void)
{
    CAGDC::Free();
    SCENG_Fini();
}


//
// NPP_New - Create a new plug-in instance.
//
NPError NP_LOADDS NPP_New(NPMIMEType /*pluginType*/, NPP pInstance, uint16 /*mode*/,
                            int16 argc, char *argn[], char *argv[],
                            NPSavedData * /*saved*/)
{
    if (pInstance == NULL)
        return NPERR_INVALID_INSTANCE_ERROR;

    CCtp *pCtp = new CCtp();
    pCtp->SetNPPInstance(pInstance);
    pInstance->pdata = pCtp;

    for (int i = 0; i < argc; i++)
    {
        if (lstrcmpi(argn[i], "Fonts") == 0)
        {
            pCtp->SetFontURL(argv[i]);
            break;
        }
    }

    return NPERR_NO_ERROR;
}


//
// NPP_Destroy - Destroy our plug-in instance.
//
NPError NP_LOADDS NPP_Destroy(NPP pInstance, NPSavedData ** /*save*/)
{
    if (pInstance == NULL)
        return NPERR_INVALID_INSTANCE_ERROR;

    CCtp *pCtp = (CCtp *)pInstance->pdata;
    if (pCtp)
    {
        HWND hWnd = pCtp->UnsubclassWindow();
        BOOL bTemp;
        pCtp->OnDestroy(0, 0, 0, bTemp);
        delete pCtp;

        HWND hParent = ::GetParent(hWnd);
        LONG lStyle = ::GetWindowLong(hParent, GWL_STYLE);
        lStyle |= WS_CLIPCHILDREN;
        ::SetWindowLong(hParent, GWL_STYLE, lStyle);
    }

    return NPERR_NO_ERROR;
}


//
// NPP_SetWindow - A window was created, resized, or destroyed.
//
NPError NP_LOADDS NPP_SetWindow(NPP pInstance, NPWindow *pNPWindow)
{
    if (pInstance == NULL)
        return NPERR_INVALID_INSTANCE_ERROR;
    CCtp *pCtp = (CCtp *)pInstance->pdata;

    if (pNPWindow == NULL || pCtp == NULL)
        return NPERR_GENERIC_ERROR;
    HWND hWnd = (HWND)(DWORD)pNPWindow->window;

    if (hWnd == NULL)
        return NPERR_NO_ERROR;

    if (hWnd != NULL && pCtp->m_hWnd == NULL)
    {
        HWND hParent = ::GetParent(hWnd);
        LONG lStyle = ::GetWindowLong(hParent, GWL_STYLE);
        lStyle &= ~WS_CLIPCHILDREN;
        ::SetWindowLong(hParent, GWL_STYLE, lStyle);

        if (! pCtp->SubclassWindow(hWnd))
        {
            delete pCtp;
            pInstance->pdata = NULL;
            return NPERR_MODULE_LOAD_FAILED_ERROR;
        }
        BOOL bTemp;
        pCtp->OnCreate(0, 0, 0, bTemp);
        ::InvalidateRect(pCtp->GetParent(), NULL, TRUE);
    }

    return NPERR_NO_ERROR;
}
```

```
//
// NPP_NewStream - A new stream was created.
//
NPError NP_LOADDS NPP_NewStream(NPP pInstance, NPMIMEType type, NPStream *stream,
                                NPBool /*seekable*/, uint16 *stype)
{
    if (pInstance == NULL)
        return NPERR_INVALID_INSTANCE_ERROR;

    CCtp *pCtp = (CCtp *)pInstance->pdata;

    if (lstrcmpi(&stream->url[lstrlen(stream->url) - 3], "ctp") == 0)
        pCtp->FileStart();
    else if (lstrcmpi(&stream->url[lstrlen(stream->url) - 3], "ttz") == 0)
        pCtp->FontStart(&stream->url[lstrlen(stream->url) - 12]);

    *stype = NP_NORMAL;

    return NPERR_NO_ERROR;
}


//
// NPP_WriteReady - Returns amount of data we can handle for the next NPP_Write
//
int32 NP_LOADDS NPP_WriteReady(NPP pInstance, NPStream * /*stream*/)
{
    if (pInstance == NULL)
        return NPERR_INVALID_INSTANCE_ERROR;

    return 0x0FFFFFFF;
}


//
// NPP_Write - Here is some data. Return -1 to abort stream.
//
int32 NP_LOADDS NPP_Write(NPP pInstance, NPStream *stream, int32 /*offset*/,
                            int32 len, void *buffer)
{
    if (pInstance == NULL)
        return NPERR_INVALID_INSTANCE_ERROR;

    CCtp *pCtp = (CCtp *)pInstance->pdata;
    if (lstrcmpi(&stream->url[lstrlen(stream->url) - 3], "ctp") == 0)
        pCtp->FileData((BYTE *)buffer, len);
    else if (lstrcmpi(&stream->url[lstrlen(stream->url) - 3], "ttz") == 0)
    {
        pCtp->FontData(&stream->url[lstrlen(stream->url) - 12],
                        (BYTE *)buffer, len);
    }

    return len;
}


//
// NPP_DestroyStream - Stream is done, but audio may still be playing.
//
NPError NP_LOADDS NPP_DestroyStream(NPP pInstance, NPStream *stream,
                                    NPError reason)
{
    if (pInstance == NULL)
        return NPERR_INVALID_INSTANCE_ERROR;

    if (reason == NPRES_DONE)
    {
        CCtp *pCtp = (CCtp *)pInstance->pdata;
        if (lstrcmpi(&stream->url[lstrlen(stream->url) - 3], "ctp") == 0)
            pCtp->FileEnd();
        else if (lstrcmpi(&stream->url[lstrlen(stream->url) - 3], "ttz") == 0)
            pCtp->FontEnd(&stream->url[lstrlen(stream->url) - 12]);
    }

    return NPERR_NO_ERROR;
}


//
// NPP_StreamAsFile - For file based plug-ins, not streaming.
//
void NP_LOADDS NPP_StreamAsFile(NPP /*pInstance*/, NPStream * /*stream*/,
                                const char * /*fname*/)
{
}


jref NP_LOADDS NPP_GetJavaClass(void)
{
    return NULL;
}


void NP_LOADDS NPP_Print(NPP /*pInstance*/, NPPrint * /*printInfo*/)
{
}


void NP_LOADDS NPP_URLNotify(NPP /* pInstance */, const char * /* url */,
  NPReason /* reason */, void * /* notifyData */)
{
}
```

```
/* npwin.cpp */

//\/ INCLUDE
#include "StdAfx.h"

// netscape
#ifndef _NPAPI_H_
#include "npapi.h"
#endif
#ifndef _NPUPP_H_
#include "npupp.h"
#endif

//\/ DEFINE
#ifdef WIN32
    #define NP_EXPORT
#else
    #define NP_EXPORT _export
#endif


//\/ GLOBAL DATA
NPNetscapeFuncs *g_pNavigatorFuncs = 0;
JRIGlobalRef    Private_GetJavaClass(void);


//\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\.
///\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/.
// Private_GetJavaClass (global function)
//
//  Given a Java class reference (thru NPP_GetJavaClass) inform JRT
//  of this class existence
//
JRIGlobalRef Private_GetJavaClass(void)
{
    jref clazz = NPP_GetJavaClass();
    if (clazz)
    {
        JRIEnv* env = NPN_GetJavaEnv();
        return JRI_NewGlobalRef(env, clazz);
    }
    return NULL;
}


//\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\.
///\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/.
//                        PLUGIN DLL entry points
//
// These are the Windows specific DLL entry points. They must be exported
//

// we need these to be global since we have to fill one of its field
// with a data (class) which requires knowlwdge of the navigator
// jump-table. This jump table is known at Initialize time (NP_Initialize)
// which is called after NP_GetEntryPoint
static NPPluginFuncs *g_pluginFuncs;


//\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\.
///\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/.
// NP_GetEntryPoints
//
//  fills in the func table used by Navigator to call entry points in
//  plugin DLL.  Note that these entry points ensure that DS is loaded
//  by using the NP_LOADDS macro, when compiling for Win16
//
NPError WINAPI NP_EXPORT NP_GetEntryPoints(NPPluginFuncs *pFuncs)
{
    // trap a NULL ptr
    if (pFuncs == NULL)
        return NPERR_INVALID_FUNCTABLE_ERROR;
    // if the plugin's function table is smaller than the plugin expects,
    // then they are incompatible, and should return an error

    pFuncs->version       = (NP_VERSION_MAJOR << 8) | NP_VERSION_MINOR;
    pFuncs->newp          = NPP_New;
    pFuncs->destroy       = NPP_Destroy;
    pFuncs->setwindow     = NPP_SetWindow;
    pFuncs->newstream     = NPP_NewStream;
    pFuncs->destroystream = NPP_DestroyStream;
    pFuncs->asfile        = NPP_StreamAsFile;
    pFuncs->writeready    = NPP_WriteReady;
    pFuncs->write         = NPP_Write;
    pFuncs->print         = NPP_Print;
    pFuncs->event         = 0;       /// reserved

    g_pluginFuncs         = pFuncs;

    return NPERR_NO_ERROR;
}


//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\.
///\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//.
// NP_Initialize
//
//  called immediately after the plugin DLL is loaded
//
NPError WINAPI NP_EXPORT NP_Initialize(NPNetscapeFuncs *pFuncs)
{
    // trap a NULL ptr
    if(pFuncs == NULL)
        return NPERR_INVALID_FUNCTABLE_ERROR;

    g_pNavigatorFuncs = pFuncs; // save it for future reference

    // if the plugin's major ver level is lower than the Navigator's,
    // then they are incompatible, and should return an error
    if (HIBYTE(pFuncs->version) > NP_VERSION_MAJOR)
        return NPERR_INCOMPATIBLE_VERSION_ERROR;

    // We have to defer these assignments until g_pNavigatorFuncs is set
    int navMinorVers = g_pNavigatorFuncs->version & 0xFF;

    if (navMinorVers >= NPVERS_HAS_NOTIFICATION)
    {
        g_pluginFuncs->urlnotify = NPP_URLNotify;
    }

#ifdef WIN32 // An ugly hack, because Win16 lags behind in Java
    if (navMinorVers >= NPVERS_HAS_LIVECONNECT) {
#else
    if (navMinorVers >= NPVERS_WIN16_HAS_LIVECONNECT) {
#endif // WIN32
        g_pluginFuncs->javaClass = Private_GetJavaClass();
    }

    // NPP_Initialize is a standard (cross-platform) initialize function.
    return NPP_Initialize();
}


//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\.
///\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//\//.
// NP_Shutdown
//
//  called immediately before the plugin DLL is unloaded.
//  This functio shuold check for some ref count on the dll to see if it is
//  unloadable or it needs to stay in memory.
//
NPError WINAPI NP_EXPORT NP_Shutdown()
{
    NPP_Shutdown();
    g_pNavigatorFuncs = NULL;
    return NPERR_NO_ERROR;
}


//                        END - PLUGIN DLL entry points
////\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/.
//\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\.

/*    NAVIGATOR Entry points    */

/* These entry points expect to be called from within the plugin.  The
   noteworthy assumption is that DS has already been set to point to the
   plugin's DLL data segment.  Don't call these functions from outside
   the plugin without ensuring DS is set to the DLLs data segment first,
   typically using the NP_LOADDS macro
*/


/* returns the major/minor version numbers of the Plugin API for the plugin
   and the Navigator
*/
void NPN_Version(int *plugin_major, int *plugin_minor, int *netscape_major,
                 int *netscape_minor)
{
    *plugin_major   = NP_VERSION_MAJOR;
    *plugin_minor   = NP_VERSION_MINOR;
    *netscape_major = HIBYTE(g_pNavigatorFuncs->version);
    *netscape_minor = LOBYTE(g_pNavigatorFuncs->version);
}


/* causes the specified URL to be fetched and streamed in
*/
NPError NPN_GetURLNotify(NPP instance, const char *url, const char *target,
                         void *notifyData)

{
    int navMinorVers = g_pNavigatorFuncs->version & 0xFF;
    NPError err;
    if (navMinorVers >= NPVERS_HAS_NOTIFICATION)
    {
        err = g_pNavigatorFuncs->geturlnotify(instance, url, target, notifyData);
    }
    else
    {
        err = NPERR_INCOMPATIBLE_VERSION_ERROR;
    }
    return err;
}


NPError NPN_GetURL(NPP instance, const char *url, const char *target)
{
    return g_pNavigatorFuncs->geturl(instance, url, target);
}


NPError NPN_PostURLNotify(NPP instance, const char *url, const char *window,
                          uint32 len, const char *buf, NPBool file, void *notifyData)
{
    int navMinorVers = g_pNavigatorFuncs->version & 0xFF;
    NPError err;
    if (navMinorVers >= NPVERS_HAS_NOTIFICATION)
    {
        err = g_pNavigatorFuncs->posturlnotify(instance, url, window, len,
                                               buf, file, notifyData);
    }
    else
    {
        err = NPERR_INCOMPATIBLE_VERSION_ERROR;
    }
    return err;
}
```

```
NPError NPN_PostURL(NPP instance, const char *url, const char *window,
                    uint32 len, const char *buf, NPBool file)
{
    return g_pNavigatorFuncs->posturl(instance, url, window, len, buf, file);
}


/* Requests that a number of bytes be provided on a stream.  Typically
   this would be used if a stream was in "pull" mode.  An optional
   position can be provided for streams which are seekable.
*/
NPError NPN_RequestRead(NPStream *stream, NPByteRange *rangeList)
{
    return g_pNavigatorFuncs->requestread(stream, rangeList);
}


/* Creates a new stream of data from the plug-in to be interpreted
   by Netscape in the current window.
*/
NPError NPN_NewStream(NPP instance, NPMIMEType type,
                        const char *target, NPStream **stream)
{
    int navMinorVersion = g_pNavigatorFuncs->version & 0xFF;
    NPError err;

    if (navMinorVersion >= NPVERS_HAS_STREAMOUTPUT)
    {
        err = g_pNavigatorFuncs->newstream(instance, type, target, stream);
    }
    else
    {
        err = NPERR_INCOMPATIBLE_VERSION_ERROR;
    }
    return err;
}


/* Provides len bytes of data.
*/
int32 NPN_Write(NPP instance, NPStream *stream,
                int32 len, void *buffer)
{
    int navMinorVersion = g_pNavigatorFuncs->version & 0xFF;
    int32 result;

    if (navMinorVersion >= NPVERS_HAS_STREAMOUTPUT)
    {
        result = g_pNavigatorFuncs->write(instance, stream, len, buffer);
    }
    else
    {
        result = -1;
    }
    return result;
}


/* Closes a stream object.
reason indicates why the stream was closed.
*/
NPError NPN_DestroyStream(NPP instance, NPStream *stream, NPError reason)
{
    int navMinorVersion = g_pNavigatorFuncs->version & 0xFF;
    NPError err;

    if (navMinorVersion >= NPVERS_HAS_STREAMOUTPUT)
    {
        err = g_pNavigatorFuncs->destroystream(instance, stream, reason);
    }
    else
    {
        err = NPERR_INCOMPATIBLE_VERSION_ERROR;
    }
    return err;
}


/* Provides a text status message in the Netscape client user interface
*/
void NPN_Status(NPP instance, const char *message)
{
    g_pNavigatorFuncs->status(instance, message);
}


/* returns the user agent string of Navigator, which contains version info
*/
const char *NPN_UserAgent(NPP instance)
{
    return g_pNavigatorFuncs->uagent(instance);
}


/* allocates memory from the Navigator's memory space.  Necessary so that
   saved instance data may be freed by Navigator when exiting.
*/
void *NPN_MemAlloc(uint32 size)
{
    return g_pNavigatorFuncs->memalloc(size);
}


/* reciprocal of MemAlloc() above
*/
void NPN_MemFree(void *ptr)
{
    g_pNavigatorFuncs->memfree(ptr);
}


/* private function to Netscape.  do not use!
*/
void NPN_ReloadPlugins(NPBool reloadPages)
{
    g_pNavigatorFuncs->reloadplugins(reloadPages);
}


JRIEnv *NPN_GetJavaEnv(void)
{
    return g_pNavigatorFuncs->getJavaEnv();
}


jref NPN_GetJavaPeer(NPP instance)
{
    return g_pNavigatorFuncs->getJavaPeer(instance);
}
```

```
#ifndef __PROP_SHEET_H__
#define __PROP_SHEET_H__

#include <commctrl.h>

template <class T>
class ATL_NO_VTABLE CPropertyPageImpl : public CDialogImplBase
{
public:
    PROPSHEETPAGE m_psp;

    operator PROPSHEETPAGE*() { return &m_psp; }

// Construction
    CPropertyPageImpl(LPCTSTR lpszTitle = NULL)
    {
        // initialize PROPSHEETPAGE struct
        memset(&m_psp, 0, sizeof(PROPSHEETPAGE));
        m_psp.dwSize = sizeof(PROPSHEETPAGE);
        m_psp.dwFlags = PSP_USECALLBACK;
        m_psp.hInstance = _Module.GetResourceInstance();
        m_psp.pszTemplate = MAKEINTRESOURCE(T::IDD);
        m_psp.pfnDlgProc = (DLGPROC)T::StartDialogProc;
        m_psp.pfnCallback = T::PropPageCallback;
        m_psp.lParam = (LPARAM)this;

        if(lpszTitle != NULL)
        {
            m_psp.pszTitle = lpszTitle;
            m_psp.dwFlags |= PSP_USETITLE;
        }
    }

    static UINT CALLBACK PropPageCallback(HWND hWnd, UINT uMsg, LPPROPSHEETPAGE ppsp)
    {
        _ASSERTE(hWnd == NULL);
        if(uMsg == PSPCB_CREATE)
        {
            CDialogImplBase* pPage = (CDialogImplBase*)ppsp->lParam;
            _Module.AddCreateWndData(&pPage->m_thunk.cd, pPage);
        }
        return 1;
    }

    HPROPSHEETPAGE Create()
    {
        return ::CreatePropertySheetPage(&m_psp);
    }

    BOOL EndDialog(int)
    {
        // do nothing here, calling ::EndDialog will close the whole sheet
        _ASSERTE(FALSE);
        return FALSE;
    }

// Operations
    void CancelToClose()
    {
        _ASSERTE(::IsWindow(m_hWnd));
        _ASSERTE(GetParent() != NULL);

        ::SendMessage(GetParent(), PSM_CANCELTOCLOSE, 0, 0L);
    }
    void SetModified(BOOL bChanged = TRUE)
    {
        _ASSERTE(::IsWindow(m_hWnd));
        _ASSERTE(GetParent() != NULL);

        if(bChanged)
            ::SendMessage(GetParent(), PSM_CHANGED, (WPARAM)m_hWnd, 0L);
        else
            ::SendMessage(GetParent(), PSM_UNCHANGED, (WPARAM)m_hWnd, 0L);
    }
    void SetWizardButtons (DWORD dwFlags)
    {
        _ASSERTE(::IsWindow(m_hWnd));
        _ASSERTE(GetParent() != NULL);

        ::PostMessage(GetParent(), PSM_SETWIZBUTTONS, 0, (LPARAM) dwFlags);
    }
    LRESULT QuerySiblings(WPARAM wParam, LPARAM lParam)
    {
        _ASSERTE(::IsWindow(m_hWnd));
        _ASSERTE(GetParent() != NULL);

        return ::SendMessage(GetParent(), PSM_QUERYSIBLINGS, wParam, lParam);
    }

    BEGIN_MSG_MAP(CPropertyPageImpl<T>)
        MESSAGE_HANDLER(WM_NOTIFY, OnNotify)
    END_MSG_MAP()

// Message handler
    LRESULT OnNotify(UINT uMsg, WPARAM wParam, LPARAM lParam, BOOL& bHandled)
    {
        _ASSERTE(::IsWindow(m_hWnd));
        NMHDR* pNMHDR = (NMHDR*)lParam;

        // don't handle messages not from the page/sheet itself
        if(pNMHDR->hwndFrom != m_hWnd && pNMHDR->hwndFrom != ::GetParent(m_hWnd))
        {
            bHandled = FALSE;
            return 1;
        }

        T* pT = (T*)this;
        LRESULT lResult = 0;
        // handle default
        switch(pNMHDR->code)
        {
        case PSN_SETACTIVE:
            lResult = pT->OnSetActive() ? 0 : -1;
            break;
        case PSN_KILLACTIVE:
            lResult = !pT->OnKillActive();
            break;
        case PSN_APPLY:
            lResult = pT->OnApply() ? PSNRET_NOERROR : PSNRET_INVALID_NOCHANGEPAGE;
            break;
        case PSN_RESET:
            pT->OnReset();
            break;
        case PSN_QUERYCANCEL:
            lResult = !pT->OnQueryCancel();
            break;
        case PSN_WIZNEXT:
            lResult = !pT->OnWizardNext();
            break;
        case PSN_WIZBACK:
            lResult = !pT->OnWizardBack();
            break;
        case PSN_WIZFINISH:
            lResult = !pT->OnWizardFinish();
            break;
        case PSN_HELP:
            lResult = pT->OnHelp();
            break;
        default:
            bHandled = FALSE;   // not handled
        }

        return lResult;
    }

// Overridables
```

```
    BOOL OnSetActive()
    {
        return TRUE;
    }
    BOOL OnKillActive()
    {
        return TRUE;
    }
    BOOL OnApply()
    {
        return TRUE;
    }
    void OnReset()
    {
    }
    BOOL OnQueryCancel()
    {
        return TRUE;    // ok to cancel
    }
    BOOL OnWizardBack()
    {
        return TRUE;
    }
    BOOL OnWizardNext()
    {
        return TRUE;
    }
    BOOL OnWizardFinish()
    {
        return TRUE;
    }
    BOOL OnHelp()
    {
        return TRUE;
    }
};

#endif //__PROP_SHEET_H__
```

```
//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by NPCTP.rc
//
#define IDD_CTLPANEL                    102
#define IDD_WAITDLG                     103
#define IDC_PAGE1                       201
#define IDR_AGLOGO                      201
#define IDC_PAGE2                       202
#define IDR_CPLOGO                      202
#define IDC_PAGE3                       203
#define IDR_CACFC                       203
#define IDC_PAGE4                       204
#define IDD_DBLSIDEINTRO                204
#define IDC_FONT                        205
#define IDD_DBLSIDESTEP1                205
#define IDC_PTSIZE                      206
#define IDD_DBLSIDEEND                  206
#define IDC_COLOR                       207
#define IDD_DBLSIDESTEP2                207
#define IDC_LEFT                        208
#define IDB_1UP                         208
#define IDC_CENTER                      209
#define IDB_2UP                         209
#define IDC_RIGHT                       210
#define IDB_3UP                         210
#define IDC_PRINT                       211
#define IDB_2DOWN                       211
#define IDC_SINGLEFOLD                  212
#define IDB_1UP2DOWN                    212
#define IDC_QUARTERFOLD                 213
#define IDC_FRAME1                      214
#define IDC_FRAME2                      215
#define IDC_FRAME3                      216
#define IDC_FRAME4                      217
#define IDC_DBLSIDE                     218

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        213
#define _APS_NEXT_COMMAND_VALUE         32768
#define _APS_NEXT_CONTROL_VALUE         219
#define _APS_NEXT_SYMED_VALUE           104
#endif
#endif
```

flamingo

cayman33

```
# Microsoft Developer Studio Project File - Name="CTPUtil" - Package Owner=<4>
# Microsoft Developer Studio Generated Build File, Format Version 5.00
# ** DO NOT EDIT **

# TARGTYPE "Win32 (x86) Application" 0x0101

CFG=CTPUtil - Win32 Debug
!MESSAGE This is not a valid makefile. To build this project using NMAKE,
!MESSAGE use the Export Makefile command and run
!MESSAGE
!MESSAGE NMAKE /f "CTPUtil.mak".
!MESSAGE
!MESSAGE You can specify a configuration when running NMAKE
!MESSAGE by defining the macro CFG on the command line. For example:
!MESSAGE
!MESSAGE NMAKE /f "CTPUtil.mak" CFG="CTPUtil - Win32 Debug"
!MESSAGE
!MESSAGE Possible choices for configuration are:
!MESSAGE
!MESSAGE "CTPUtil - Win32 Release" (based on "Win32 (x86) Application")
!MESSAGE "CTPUtil - Win32 Debug" (based on "Win32 (x86) Application")
!MESSAGE

# Begin Project
# PROP Scc_ProjName ""$/CTPUtil", OKAAAAAA"
# PROP Scc_LocalPath "."
CPP=cl.exe
MTL=midl.exe
RSC=rc.exe

!IF  "$(CFG)" == "CTPUtil - Win32 Release"

# PROP BASE Use_MFC 5
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "Release"
# PROP BASE Intermediate_Dir "Release"
# PROP BASE Target_Dir ""
# PROP Use_MFC 5
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Release"
# PROP Intermediate_Dir "Release"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MT /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_WINDOWS" /Yu"stdafx.h" /FD /c
# ADD CPP /nologo /Zp2 /MT /W3 /GX /O2 /I "..\Stonehnd" /I "..\ZLib" /D "WIN32" /D "NDEBUG" /D "_WIN
DOWS" /Yu"stdafx.h" /FD /c
# ADD BASE MTL /nologo /D "NDEBUG" /mktyplib203 /o NUL /win32
# ADD MTL /nologo /D "NDEBUG" /mktyplib203 /o NUL /win32
# ADD BASE RSC /l 0x409 /d "NDEBUG"
# ADD RSC /l 0x409 /d "NDEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 /nologo /subsystem:windows /machine:I386
# ADD LINK32 version.lib /nologo /subsystem:windows /machine:I386

!ELSEIF  "$(CFG)" == "CTPUtil - Win32 Debug"

# PROP BASE Use_MFC 5
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "Debug"
# PROP BASE Intermediate_Dir "Debug"
# PROP BASE Target_Dir ""
# PROP Use_MFC 5
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Debug"
# PROP Intermediate_Dir "Debug"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MTd /W3 /Gm /GX /Zi /Od /D "WIN32" /D "_DEBUG" /D "_WINDOWS" /Yu"stdafx.h" /
FD /c
# ADD CPP /nologo /Zp2 /MTd /W3 /Gm /GX /Zi /Od /I "..\Stonehnd" /I "..\ZLib" /D "WIN32" /D "_DEBUG"
```

```
 /D "_WINDOWS" /Yu"stdafx.h" /FD /c
# ADD BASE MTL /nologo /D "_DEBUG" /mktyplib203 /o NUL /win32
# ADD MTL /nologo /D "_DEBUG" /mktyplib203 /o NUL /win32
# ADD BASE RSC /l 0x409 /d "_DEBUG"
# ADD RSC /l 0x409 /d "_DEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 /nologo /subsystem:windows /debug /machine:I386 /pdbtype:sept
# ADD LINK32 version.lib /nologo /subsystem:windows /debug /machine:I386 /pdbtype:sept

!ENDIF

# Begin Target

# Name "CTPUtil - Win32 Release"
# Name "CTPUtil - Win32 Debug"
# Begin Group "Source Files"

# PROP Default_Filter "cpp;c;cxx;rc;def;r;odl;idl;hpj;bat"
# Begin Source File

SOURCE=.\AGDC.cpp
# End Source File
# Begin Source File

SOURCE=.\AGDoc.cpp
# End Source File
# Begin Source File

SOURCE=.\AGLayer.cpp
# End Source File
# Begin Source File

SOURCE=.\AGMatrix.cpp
# End Source File
# Begin Source File

SOURCE=.\AGPage.cpp
# End Source File
# Begin Source File

SOURCE=.\AGSym.cpp
# End Source File
# Begin Source File

SOURCE=.\AGText.cpp
# End Source File
# Begin Source File

SOURCE=.\CTPUtil.cpp
# End Source File
# Begin Source File

SOURCE=.\CTPUtil.rc

!IF  "$(CFG)" == "CTPUtil - Win32 Release"

!ELSEIF  "$(CFG)" == "CTPUtil - Win32 Debug"

!ENDIF

# End Source File
# Begin Source File

SOURCE=.\MainFrm.cpp
# End Source File
# Begin Source File

SOURCE=.\ProgDlg.cpp
# End Source File
# Begin Source File
```

```
SOURCE=.\StdAfx.cpp
# ADD CPP /Yc"stdafx.h"
# End Source File
# End Group
# Begin Group "Header Files"

# PROP Default_Filter "h;hpp;hxx;hm;inl"
# Begin Source File

SOURCE=.\AGDC.h
# End Source File
# Begin Source File

SOURCE=.\AGDib.h
# End Source File
# Begin Source File

SOURCE=.\AGDoc.h
# End Source File
# Begin Source File

SOURCE=.\AGLayer.h
# End Source File
# Begin Source File

SOURCE=.\AGMatrix.h
# End Source File
# Begin Source File

SOURCE=.\AGPage.h
# End Source File
# Begin Source File

SOURCE=.\AGSym.h
# End Source File
# Begin Source File

SOURCE=.\AGText.h
# End Source File
# Begin Source File

SOURCE=.\CTPUtil.h
# End Source File
# Begin Source File

SOURCE=.\MainFrm.h
# End Source File
# Begin Source File

SOURCE=.\ProgDlg.h
# End Source File
# Begin Source File

SOURCE=.\Resource.h
# End Source File
# Begin Source File

SOURCE=.\StdAfx.h
# End Source File
# Begin Source File

SOURCE=.\version.h
# End Source File
# End Group
# Begin Group "Resource Files"

# PROP Default_Filter "ico;cur;bmp;dlg;rc2;rct;bin;cnt;rtf;gif;jpg;jpeg;jpe"
# Begin Source File

SOURCE=.\res\CTPUtil.ico
# End Source File
# Begin Source File
```

```
SOURCE=.\res\CTPUtil.rc2
# End Source File
# End Group
# End Target
# End Project
```

```
//
// CTPUtil.cpp : Defines the class behaviors for the application.
//

#include "stdafx.h"
#include "CTPUtil.h"

#include "MainFrm.h"
#include "scappint.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif


/////////////////////////////////////////////////////////////////////////////
// CCTPUtilApp

BEGIN_MESSAGE_MAP(CCTPUtilApp, CWinApp)
    //{{AFX_MSG_MAP(CCTPUtilApp)
    ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
    ON_COMMAND(ID_FILE_PRINT_SETUP, OnFilePrintSetup)
    //}}AFX_MSG_MAP
    // Standard file based document commands
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CCTPUtilApp construction

CCTPUtilApp::CCTPUtilApp()
{
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CCTPUtilApp object

CCTPUtilApp theApp;


/////////////////////////////////////////////////////////////////////////////
// CCTPUtilApp initialization

BOOL CCTPUtilApp::InitInstance()
{
    // Standard initialization

#ifdef _AFXDLL
    Enable3dControls();         // Call this when using MFC in a shared DLL
#else
    Enable3dControlsStatic();   // Call this when linking to MFC statically
#endif

    SCENG_Init();

    m_pMainWnd = new CMainFrame();
    m_pMainWnd->ShowWindow(SW_SHOW);
    m_pMainWnd->UpdateWindow();

    return TRUE;
}


/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About
//
class CAboutDlg : public CDialog
{
public:
    CAboutDlg();
```

```
// Dialog Data
    //{{AFX_DATA(CAboutDlg)
    enum { IDD = IDD_ABOUTBOX };
    CString m_csVersion;
    //}}AFX_DATA

    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CAboutDlg)
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:
    //{{AFX_MSG(CAboutDlg)
    virtual BOOL OnInitDialog();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
    //{{AFX_DATA_INIT(CAboutDlg)
    m_csVersion = _T("");
    //}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CAboutDlg)
    DDX_Text(pDX, IDC_VERSION, m_csVersion);
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
    //{{AFX_MSG_MAP(CAboutDlg)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

// App command to run the dialog
void CCTPUtilApp::OnAppAbout()
{
    CAboutDlg aboutDlg;
    aboutDlg.DoModal();
}

/////////////////////////////////////////////////////////////////////////////
// CCTPUtilApp commands

BOOL CAboutDlg::OnInitDialog()
{
    char    szAppPath[_MAX_PATH];
    DWORD   dwVerInfoSize;
    DWORD   dwVerHnd;
    UINT    uVersionLen;
    LPSTR   lpVersion;
    CString csVersion;

    csVersion.Empty();

    ::GetModuleFileName(AfxGetInstanceHandle(), szAppPath, sizeof (szAppPath));
    dwVerInfoSize = ::GetFileVersionInfoSize(szAppPath, &dwVerHnd);
    if (dwVerInfoSize)
    {
        LPSTR lpVffInfo = (LPSTR)malloc((int)dwVerInfoSize);
        if (lpVffInfo)
        {
            ::GetFileVersionInfo (szAppPath, dwVerHnd, dwVerInfoSize, lpVffInfo);
            ::VerQueryValue (lpVffInfo,
              "\\StringFileInfo\\040904B0\\ProductVersion",
              (LPVOID *)&lpVersion, &uVersionLen);
            if (uVersionLen)
                csVersion = lpVersion;
            free (lpVffInfo);
        }
    }

    UpdateData (TRUE);
    m_csVersion += csVersion;

    CDialog::OnInitDialog();

    return TRUE;  // return TRUE unless you set the focus to a control
                  // EXCEPTION: OCX Property Pages should return FALSE
}


int CCTPUtilApp::ExitInstance()
{
    SCENG_Fini();

    return CWinApp::ExitInstance();
}
```

```
# Microsoft Developer Studio Project File - Name="CTPUtil" - Packa
ge Owner=<4>
# Microsoft Developer Studio Generated Build File, Format Version
6.00
# ** DO NOT EDIT **

# TARGTYPE "Win32 (x86) Application" 0x0101

CFG=CTPUtil - Win32 Debug
!MESSAGE This is not a valid makefile. To build this project using
 NMAKE,
!MESSAGE use the Export Makefile command and run
!MESSAGE
!MESSAGE NMAKE /f "CTPUtil.mak".
!MESSAGE
!MESSAGE You can specify a configuration when running NMAKE
!MESSAGE by defining the macro CFG on the command line. For exampl
e:
!MESSAGE
!MESSAGE NMAKE /f "CTPUtil.mak" CFG="CTPUtil - Win32 Debug"
!MESSAGE
!MESSAGE Possible choices for configuration are:
!MESSAGE
!MESSAGE "CTPUtil - Win32 Release" (based on "Win32 (x86) Applicat
ion")
!MESSAGE "CTPUtil - Win32 Debug" (based on "Win32 (x86) Applicatio
n")
!MESSAGE

# Begin Project
# PROP AllowPerConfigDependencies 0
# PROP Scc_ProjName ""$/CTPUtil", OKAAAAAA"
# PROP Scc_LocalPath "."
CPP=cl.exe
MTL=midl.exe
RSC=rc.exe

!IF  "$(CFG)" == "CTPUtil - Win32 Release"

# PROP BASE Use_MFC 5
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "Release"
# PROP BASE Intermediate_Dir "Release"
# PROP BASE Target_Dir ""
# PROP Use_MFC 5
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Release"
# PROP Intermediate_Dir "Release"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MT /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "
_WINDOWS" /Yu"stdafx.h" /FD /c
# ADD CPP /nologo /Zp2 /MT /W3 /GX /O2 /I "..\Stonehnd" /I "..\ZLi
b" /D "WIN32" /D "NDEBUG" /D "_WINDOWS" /Yu"stdafx.h" /FD /c
# ADD BASE MTL /nologo /D "NDEBUG" /mktyplib203 /o "NUL" /win32
# ADD MTL /nologo /D "NDEBUG" /mktyplib203 /o "NUL" /win32
# ADD BASE RSC /l 0x409 /d "NDEBUG"
# ADD RSC /l 0x409 /d "NDEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 /nologo /subsystem:windows /machine:I386
# ADD LINK32 version.lib /nologo /subsystem:windows /machine:I386

!ELSEIF  "$(CFG)" == "CTPUtil - Win32 Debug"

# PROP BASE Use_MFC 5
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "Debug"
# PROP BASE Intermediate_Dir "Debug"
# PROP BASE Target_Dir ""
# PROP Use_MFC 5
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Debug"
# PROP Intermediate_Dir "Debug"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /MTd /W3 /Gm /GX /Zi /Od /D "WIN32" /D "_DE
BUG" /D "_WINDOWS" /Yu"stdafx.h" /FD /c
# ADD CPP /nologo /Zp2 /MTd /W3 /Gm /GX /ZI /Od /I "..\Stonehnd" /
I "..\ZLib" /D "WIN32" /D "_DEBUG" /D "_WINDOWS" /Yu"stdafx.h" /FD
 /c
# ADD BASE MTL /nologo /D "_DEBUG" /mktyplib203 /o "NUL" /win32
# ADD MTL /nologo /D "_DEBUG" /mktyplib203 /o "NUL" /win32
# ADD BASE RSC /l 0x409 /d "_DEBUG"
# ADD RSC /l 0x409 /d "_DEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 /nologo /subsystem:windows /debug /machine:I386
```

```
/pdbtype:sept
# ADD LINK32 version.lib /nologo /subsystem:windows /debug /machin
e:I386 /pdbtype:sept

!ENDIF

# Begin Target

# Name "CTPUtil - Win32 Release"
# Name "CTPUtil - Win32 Debug"
# Begin Group "Source Files"

# PROP Default_Filter "cpp;c;cxx;rc;def;r;odl;idl;hpj;bat"
# Begin Source File

SOURCE=.\AGDC.cpp
# End Source File
# Begin Source File

SOURCE=.\AGDoc.cpp
# End Source File
# Begin Source File

SOURCE=.\AGLayer.cpp
# End Source File
# Begin Source File

SOURCE=.\AGMatrix.cpp
# End Source File
# Begin Source File

SOURCE=.\AGPage.cpp
# End Source File
# Begin Source File

SOURCE=.\AGSym.cpp
# End Source File
# Begin Source File

SOURCE=.\AGText.cpp
# End Source File
# Begin Source File

SOURCE=.\CTPUtil.cpp
# End Source File
# Begin Source File
```

```
SOURCE=.\CTPUtil.rc
# End Source File
# Begin Source File

SOURCE=.\MainFrm.cpp
# End Source File
# Begin Source File

SOURCE=.\ProgDlg.cpp
# End Source File
# Begin Source File

SOURCE=.\StdAfx.cpp
# ADD CPP /Yc"stdafx.h"
# End Source File
# End Group
# Begin Group "Header Files"

# PROP Default_Filter "h;hpp;hxx;hm;inl"
# Begin Source File

SOURCE=.\AGDC.h
# End Source File
# Begin Source File

SOURCE=.\AGDib.h
# End Source File
# Begin Source File

SOURCE=.\AGDoc.h
# End Source File
# Begin Source File

SOURCE=.\AGLayer.h
# End Source File
# Begin Source File

SOURCE=.\AGMatrix.h
# End Source File
# Begin Source File

SOURCE=.\AGPage.h
# End Source File
# Begin Source File
```

```
SOURCE=.\AGSym.h
# End Source File
# Begin Source File

SOURCE=.\AGText.h
# End Source File
# Begin Source File

SOURCE=.\CTPUtil.h
# End Source File
# Begin Source File

SOURCE=.\MainFrm.h
# End Source File
# Begin Source File

SOURCE=.\ProgDlg.h
# End Source File
# Begin Source File

SOURCE=.\Resource.h
# End Source File
# Begin Source File

SOURCE=.\StdAfx.h
# End Source File
# Begin Source File

SOURCE=.\version.h
# End Source File
# End Group
# Begin Group "Resource Files"

# PROP Default_Filter "ico;cur;bmp;dlg;rc2;rct;bin;cnt;rtf;gif;jpg
;jpeg;jpe"
# Begin Source File

SOURCE=.\res\CTPUtil.ico
# End Source File
# Begin Source File

SOURCE=.\res\CTPUtil.rc2
# End Source File
# End Group
# End Target
# End Project
```

```
Microsoft Developer Studio Workspace File, Format Version 6.00
# WARNING: DO NOT EDIT OR DELETE THIS WORKSPACE FILE!

###################################################################
#############

Project: "CTPUtil"=.\CTPUtil.dsp - Package Owner=<4>

Package=<5>
{{{
    begin source code control
    "$/CTPUtil", OKAAAAAA
    .
    end source code control
}}}

Package=<4>
{{{
    Begin Project Dependency
    Project_Dep_Name ZLib
    End Project Dependency
    Begin Project Dependency
    Project_Dep_Name Stonehnd
    End Project Dependency
}}}

###################################################################
#############

Project: "Stonehnd"=..\Stonehnd\Stonehnd.dsp - Package Owner=<4>

Package=<5>
{{{
    begin source code control
    "$/Stonehnd", CGAAAAAA
    ..\stonehnd
    end source code control
}}}

Package=<4>
{{{
}}}

###################################################################
#############
```

```
Project: "ZLib"=..\ZLib\ZLib.dsp - Package Owner=<4>

Package=<5>
{{{
    begin source code control
    "$/ZLib", LBAAAAAA
    ..\zlib
    end source code control
}}}

Package=<4>
{{{
}}}

#################################################################
#############

Global:

Package=<5>
{{{
    begin source code control
    "$/CTPUtil", OKAAAAAA
    .
    end source code control
}}}

Package=<3>
{{{
}}}

#################################################################
#############
```

```
// CTPUtil.h : main header file for the CTPUTIL application
//

#if !defined(AFX_CTPUTIL_H__64AE5205_3509_11D3_9331_0080C6F796A1__INCLUDED_)
#define AFX_CTPUTIL_H__64AE5205_3509_11D3_9331_0080C6F796A1__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

#ifndef __AFXWIN_H__
    #error include 'stdafx.h' before including this file for PCH
#endif

#include "resource.h"       // main symbols

/////////////////////////////////////////////////////////////////////////////
// CCTPUtilApp:
// See CTPUtil.cpp for the implementation of this class
//

class CCTPUtilApp : public CWinApp
{
public:
    CCTPUtilApp();

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CCTPUtilApp)
    public:
    virtual BOOL InitInstance();
    virtual int ExitInstance();
    //}}AFX_VIRTUAL

// Implementation

    //{{AFX_MSG(CCTPUtilApp)
    afx_msg void OnAppAbout();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};


/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately before the previous li
ne.

#endif // !defined(AFX_CTPUTIL_H__64AE5205_3509_11D3_9331_0080C6F796A1__INCLUDED_)
```

```
<html>
<body>
<pre>
<h1>Build Log</h1>
<h3>
--------------------Configuration: CTPUtil - Win32 Debug--------------------
</h3>
<h3>Command Lines</h3>
Creating command line "rc.exe /l 0x409 /fo"Debug/CTPUtil.res" /d "_DEBUG" "C:\Work\CrtPrt\CTPUtil\CT
PUtil.rc""
Creating temporary file "c:\windows\TEMP\RSP8372.TMP" with contents
[
/nologo /Zp2 /MTd /W3 /Gm /GX /ZI /Od /I "..\Stonehnd" /I "..\ZLib" /D "WIN32" /D "_DEBUG" /D "_WIND
OWS" /Fp"Debug/CTPUtil.pch" /Yu"stdafx.h" /Fo"Debug/" /Fd"Debug/" /FD /c
"C:\Work\CrtPrt\CTPUtil\AGDC.cpp"
"C:\Work\CrtPrt\CTPUtil\AGDoc.cpp"
"C:\Work\CrtPrt\CTPUtil\AGLayer.cpp"
"C:\Work\CrtPrt\CTPUtil\AGMatrix.cpp"
"C:\Work\CrtPrt\CTPUtil\AGPage.cpp"
"C:\Work\CrtPrt\CTPUtil\AGSym.cpp"
"C:\Work\CrtPrt\CTPUtil\AGText.cpp"
"C:\Work\CrtPrt\CTPUtil\CTPUtil.cpp"
"C:\Work\CrtPrt\CTPUtil\MainFrm.cpp"
"C:\Work\CrtPrt\CTPUtil\ProgDlg.cpp"
]
Creating command line "cl.exe @c:\windows\TEMP\RSP8372.TMP"
Creating temporary file "c:\windows\TEMP\RSP8373.TMP" with contents
[
/nologo /Zp2 /MTd /W3 /Gm /GX /ZI /Od /I "..\Stonehnd" /I "..\ZLib" /D "WIN32" /D "_DEBUG" /D "_WIND
OWS" /Fp"Debug/CTPUtil.pch" /Yc"stdafx.h" /Fo"Debug/" /Fd"Debug/" /FD /c
"C:\Work\CrtPrt\CTPUtil\StdAfx.cpp"
]
Creating command line "cl.exe @c:\windows\TEMP\RSP8373.TMP"
Creating temporary file "c:\windows\TEMP\RSP8374.TMP" with contents
[
version.lib /nologo /subsystem:windows /incremental:yes /pdb:"Debug/CTPUtil.pdb" /debug /machine:I38
6 /out:"Debug/CTPUtil.exe" /pdbtype:sept
.\Debug\AGDC.obj
.\Debug\AGDoc.obj
.\Debug\AGLayer.obj
.\Debug\AGMatrix.obj
.\Debug\AGPage.obj
.\Debug\AGSym.obj
.\Debug\AGText.obj
.\Debug\CTPUtil.obj
.\Debug\MainFrm.obj
.\Debug\ProgDlg.obj
.\Debug\StdAfx.obj
.\Debug\CTPUtil.res
\Work\CrtPrt\ZLib\Debug\ZLib.lib
\Work\CrtPrt\Stonehnd\Debug\Stonehnd.lib
]
Creating command line "link.exe @c:\windows\TEMP\RSP8374.TMP"
<h3>Output Window</h3>
Compiling resources...
Compiling...
StdAfx.cpp
Compiling...
AGDC.cpp
AGDoc.cpp
AGLayer.cpp
AGMatrix.cpp
AGPage.cpp
AGSym.cpp
AGText.cpp
CTPUtil.cpp
MainFrm.cpp
ProgDlg.cpp
Generating Code...
Linking...
```

```
<h3>Results</h3>
CTPUtil.exe - 0 error(s), 0 warning(s)
</pre>
</body>
</html>
```

09629370.073400

```
//Microsoft Developer Studio generated resource script.
//
#include "resource.h"

#define APSTUDIO_READONLY_SYMBOLS
//////////////////////////////////////////////////////////////////
///////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"

//////////////////////////////////////////////////////////////////
///////////
#undef APSTUDIO_READONLY_SYMBOLS

//////////////////////////////////////////////////////////////////
///////////
// English (U.S.) resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
#pragma code_page(1252)
#endif //_WIN32

#ifdef APSTUDIO_INVOKED
//////////////////////////////////////////////////////////////////
///////////
//
// TEXTINCLUDE
//

1 TEXTINCLUDE DISCARDABLE
BEGIN
    "resource.h\0"
END

2 TEXTINCLUDE DISCARDABLE
BEGIN
    "#include ""afxres.h""\r\n"
    "\0"
END

3 TEXTINCLUDE DISCARDABLE
BEGIN
```

```
    "#define _AFX_NO_SPLITTER_RESOURCES\r\n"
    "#define _AFX_NO_OLE_RESOURCES\r\n"
    "#define _AFX_NO_TRACKER_RESOURCES\r\n"
    "#define _AFX_NO_PROPERTY_RESOURCES\r\n"
    "\r\n"
    "#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)\r\n"
    "#ifdef _WIN32\r\n"
    "LANGUAGE 9, 1\r\n"
    "#pragma code_page(1252)\r\n"
    "#endif\r\n"
    "#include ""res\\CTPUtil.rc2""  // non-Microsoft Visual C++ ed
ited resources\r\n"
    "#include ""afxres.rc""         // Standard components\r\n"
    "#endif\0"
END

#endif    // APSTUDIO_INVOKED


/////////////////////////////////////////////////////////////////
///////////
//
// Icon
//

// Icon with lowest ID value placed first to ensure application ic
on
// remains consistent on all systems.
IDR_MAINFRAME           ICON    DISCARDABLE     "res\\CTPUtil.ico"

/////////////////////////////////////////////////////////////////
///////////
//
// Menu
//

IDR_MAINFRAME MENU PRELOAD DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "Batch &Print...",             ID_BATCHPRINT
        MENUITEM "Build &Thumbs...",            ID_BUILDTHUMBS
        MENUITEM "&Card List...",               ID_CARDLIST
        MENUITEM SEPARATOR
        MENUITEM "P&rinter Setup...",           ID_FILE_PRINT_SETU
P
```

```
        MENUITEM SEPARATOR
        MENUITEM "E&xit",                           ID_APP_EXIT
    END
    POPUP "Print &Options"
    BEGIN
        MENUITEM "&Default Format Only",            ID_PRINTDEFAULT
        MENUITEM "Force to &Quarter-fold",          ID_PRINTQUARTER
        MENUITEM "Force to &Single-fold",           ID_PRINTSINGLE
        MENUITEM "Print &Both formats",             ID_PRINTBOTH
    END
    POPUP "&Help"
    BEGIN
        MENUITEM "&About CTPUtil...",               ID_APP_ABOUT
    END
END


/////////////////////////////////////////////////////////////////////
///////////
//
// Dialog
//

IDD_ABOUTBOX DIALOG DISCARDABLE  0, 0, 217, 65
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "About CTPUtil"
FONT 8, "MS Sans Serif"
BEGIN
    CTEXT           "CTPUtil Version ",IDC_VERSION,5,10,208,8,SS_N
OPREFIX
    DEFPUSHBUTTON   "OK",IDOK,84,39,50,14,WS_GROUP
END

CG_IDD_PROGRESS DIALOG DISCARDABLE  0, 0, 271, 79
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSM
ENU
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON   "Cancel",IDCANCEL,110,58,50,14
    CONTROL         "Progress1",CG_IDC_PROGDLG_PROGRESS,"msctls_pr
ogress32",
                    WS_BORDER,17,34,236,13
    LTEXT           " 0 %",CG_IDC_PROGDLG_PERCENT,126,23,18,8
    LTEXT           "",CG_IDC_PROGDLG_STATUS,13,7,242,8
END
```

```
//////////////////////////////////////////////////////////////////////
///////////
//
// DESIGNINFO
//

#ifdef APSTUDIO_INVOKED
GUIDELINES DESIGNINFO DISCARDABLE
BEGIN
    IDD_ABOUTBOX, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 210
        TOPMARGIN, 7
        BOTTOMMARGIN, 58
    END

    CG_IDD_PROGRESS, DIALOG
    BEGIN
        LEFTMARGIN, 7
        RIGHTMARGIN, 264
        TOPMARGIN, 7
        BOTTOMMARGIN, 72
    END
END
#endif    // APSTUDIO_INVOKED


//////////////////////////////////////////////////////////////////////
///////////
//
// String Table
//

STRINGTABLE DISCARDABLE
BEGIN
    CG_IDS_PROGRESS_CAPTION "Printing..."
    IDS_PROGRESS_THUMBS     "Building Thumbs..."
    IDS_PROGRESS_CARDLIST   "Generating Card List..."
END

#endif    // English (U.S.) resources
//////////////////////////////////////////////////////////////////////
///////////
```

```
#ifndef APSTUDIO_INVOKED
//////////////////////////////////////////////////////////////////////
///////////
//
// Generated from the TEXTINCLUDE 3 resource.
//
#define _AFX_NO_SPLITTER_RESOURCES
#define _AFX_NO_OLE_RESOURCES
#define _AFX_NO_TRACKER_RESOURCES
#define _AFX_NO_PROPERTY_RESOURCES

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE 9, 1
#pragma code_page(1252)
#endif
#include "res\CTPUtil.rc2"  // non-Microsoft Visual C++ edited res
ources
#include "afxres.rc"         // Standard components
#endif
//////////////////////////////////////////////////////////////////////
///////////
#endif    // not APSTUDIO_INVOKED
```

```
// MainFrm.cpp : implementation of the CMainFrame class
//

#include "stdafx.h"
#include "CTPUtil.h"
#include "MainFrm.h"
#include "AGDoc.h"
#include "AGDib.h"
#include "ProgDlg.h"

#include <fstream.h>
#include <windowsx.h>
#include <math.h>

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif


#define MAX_FILEBUF 65535

typedef struct {
    int xsize;          /* horizontal size of the image in Pixels */
    int ysize;          /* vertical size of the image in Pixels */
    BYTE * data;        /* pointer to first scanline of image */
    int span;           /* byte offset between two scanlines */
} Image;

void zoom (Image *dst, Image *src);


/////////////////////////////////////////////////////////////////////////////
// CMainFrame

3EGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
    //{{AFX_MSG_MAP(CMainFrame)
    ON_COMMAND(ID_BATCHPRINT, OnBatchPrint)
    ON_COMMAND(ID_BUILDTHUMBS, OnBuildThumbs)
    ON_UPDATE_COMMAND_UI(ID_PRINTBOTH, OnUpdatePrintBoth)
    ON_COMMAND(ID_PRINTBOTH, OnPrintOptions)
    ON_UPDATE_COMMAND_UI(ID_PRINTDEFAULT, OnUpdatePrintDefault)
    ON_UPDATE_COMMAND_UI(ID_PRINTQUARTER, OnUpdatePrintQuarter)
    ON_UPDATE_COMMAND_UI(ID_PRINTSINGLE, OnUpdatePrintSingle)
    ON_COMMAND(ID_PRINTDEFAULT, OnPrintOptions)
    ON_COMMAND(ID_PRINTQUARTER, OnPrintOptions)
    ON_COMMAND(ID_PRINTSINGLE, OnPrintOptions)
    ON_COMMAND(ID_CARDLIST, OnCardList)
    //}}AFX_MSG_MAP
 ND_MESSAGE_MAP()

 ////////////////////////////////////////////////////////////////////////////
 / CMainFrame construction/destruction

 MainFrame::CMainFrame()

    Create (NULL, "Create and Print Utility", WS_OVERLAPPEDWINDOW,
      rectDefault, NULL, MAKEINTRESOURCE (IDR_MAINFRAME));

    m_nPrintOption = ID_PRINTDEFAULT;


 MainFrame::~CMainFrame()



 OOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)

    return CFrameWnd::PreCreateWindow(cs);


 ////////////////////////////////////////////////////////////////////////////
```

```
// CMainFrame diagnostics

#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
    CFrameWnd::AssertValid();
}

void CMainFrame::Dump(CDumpContext& dc) const
{
    CFrameWnd::Dump(dc);
}

#endif //_DEBUG


/////////////////////////////////////////////////////////////////////////////
// CMainFrame message handlers

void CMainFrame::OnBatchPrint()
{
    CString csFilters;

    csFilters = "Create and Print Files (*.ctp)";
    csFilters += "|";
    csFilters += "*.ctp";
    csFilters += "|";
    csFilters += "|";

    CFileDialog FileDlg (TRUE, NULL, NULL,
      OFN_HIDEREADONLY | OFN_FILEMUSTEXIST | OFN_ALLOWMULTISELECT,
      csFilters, AfxGetMainWnd ());
    FileDlg.m_ofn.lpstrTitle = "Select File(s) to Print";
    FileDlg.m_ofn.lpstrFile = (char *) malloc (MAX_FILEBUF);
    *FileDlg.m_ofn.lpstrFile = 0;
    FileDlg.m_ofn.nMaxFile = MAX_FILEBUF;

    if (FileDlg.DoModal () == IDOK)
    {
        PRINTDLG PrintDlg;
        if (AfxGetApp ()->GetPrinterDeviceDefaults (&PrintDlg))
        {
            DEVNAMES *pDevNames = (DEVNAMES *) GlobalLock (PrintDlg.hDevNames);
            DEVMODE *pDevMode = (DEVMODE *) GlobalLock (PrintDlg.hDevMode);
            char *pszDriver = ((char *) pDevNames) + pDevNames->wDriverOffset;
            char *pszDevice = ((char *) pDevNames) + pDevNames->wDeviceOffset;
            char *pszOutput = ((char *) pDevNames) + pDevNames->wOutputOffset;

            CProgressDlg ProgDlg;
            ProgDlg.Create();

            int nFileCount = 0;
            POSITION Pos = FileDlg.GetStartPosition ();
            while (Pos != NULL)
            {
                nFileCount++;
                FileDlg.GetNextPathName (Pos);
            }
            ProgDlg.SetRange (0, nFileCount);

            Pos = FileDlg.GetStartPosition ();
            while (Pos != NULL)
            {
                CString csFileName = FileDlg.GetNextPathName (Pos);
                CString csMsg = "Printing ";
                csMsg += csFileName;
                ProgDlg.SetStatus (csMsg);
                if (ProgDlg.CheckCancelButton ())
                    break;

                CAGDoc *pAGDoc = new CAGDoc ();
                ifstream In (csFileName, ios::in | ios::nocreate | ios::binary);
                if (pAGDoc->Read (In))
                {
                    int nOption = m_nPrintOption;
                    if (nOption == ID_PRINTDEFAULT)
                    {
                        AGDOCTYPE DocType = pAGDoc->GetDocType ();
                        if (DocType == DOC_CARDFV || DocType == DOC_CARDFH)
                            nOption = ID_PRINTQUARTER;
                        else if (DocType == DOC_CARDHV || DocType == DOC_CARDHH)
                            nOption = ID_PRINTSINGLE;
                        else
                            nOption = 0;
                    }

                    if (ProgDlg.CheckCancelButton ())
                        break;

                    if (nOption == ID_PRINTQUARTER || nOption == ID_PRINTBOTH)
                    {
                        pAGDoc->PrintCardQuarter (pszDriver, pszDevice,
                          pszOutput, pDevMode, csFileName);
                    }

                    if (ProgDlg.CheckCancelButton ())
                        break;

                    if (nOption == ID_PRINTSINGLE || nOption == ID_PRINTBOTH)
                    {
                        bool bRotated;
                        pAGDoc->PrintCardSingle (PRINT_BOTH, pszDriver,
                          pszDevice, pszOutput, pDevMode, bRotated, csFileName);
                    }

                    ProgDlg.StepIt ();
                    if (ProgDlg.CheckCancelButton ())
                        break;
                }
                delete pAGDoc;

            }

            ::GlobalUnlock (PrintDlg.hDevMode);
            ::GlobalUnlock (PrintDlg.hDevNames);
        }
    }
    free (FileDlg.m_ofn.lpstrFile);
}

void CMainFrame::OnBuildThumbs()
{
    CString csFilters;

    csFilters = "Create and Print Files (*.ctp)";
    csFilters += "|";
    csFilters += "*.ctp";
    csFilters += "|";
    csFilters += "|";

    CFileDialog FileDlg (TRUE, NULL, NULL,
      OFN_HIDEREADONLY | OFN_FILEMUSTEXIST | OFN_ALLOWMULTISELECT,
      csFilters, AfxGetMainWnd ());
    FileDlg.m_ofn.lpstrTitle = "Select File(s) to Build Thumbs";
    FileDlg.m_ofn.lpstrFile = (char *) malloc (MAX_FILEBUF);
    *FileDlg.m_ofn.lpstrFile = 0;
    FileDlg.m_ofn.nMaxFile = MAX_FILEBUF;

    if (FileDlg.DoModal () == IDOK)
    {
        CProgressDlg ProgDlg (IDS_PROGRESS_THUMBS);
        ProgDlg.Create();

        int nFileCount = 0;
        POSITION Pos = FileDlg.GetStartPosition ();
        while (Pos != NULL)
        {
            nFileCount++;
            FileDlg.GetNextPathName (Pos);
        }
        ProgDlg.SetRange (0, nFileCount);

        Pos = FileDlg.GetStartPosition ();
        while (Pos != NULL)
        {
            CString csFileName = FileDlg.GetNextPathName (Pos);
            CString csMsg = "Building Thumbs for ";
            csMsg += csFileName;
            ProgDlg.SetStatus (csMsg);
            if (ProgDlg.CheckCancelButton ())
                break;

            CAGDoc *pAGDoc = new CAGDoc ();
            ifstream In (csFileName, ios::in | ios::nocreate | ios::binary);

            if (pAGDoc->Read (In))
            {
                CAGPage *pPage = pAGDoc->GetPage (1);

                int nLast = csFileName.ReverseFind ('\\');
                CString csThumbT = csFileName.Left (nLast) + "\\..\\Thumb" +
                  csFileName.Mid (nLast, csFileName.GetLength () - nLast - 5) +
                  "T.bmp";
                CString csThumbD = csFileName.Left (nLast) + "\\..\\Detail" +
                  csFileName.Mid (nLast, csFileName.GetLength () - nLast - 5) +
                  "D.bmp";

                if (ProgDlg.CheckCancelButton ())
                    break;

                CreateThumb (pPage, 153*2, csThumbT);

                if (ProgDlg.CheckCancelButton ())
                    break;

                CreateThumb (pPage, 400*2, csThumbD);

                ProgDlg.StepIt ();
                if (ProgDlg.CheckCancelButton ())
                    break;
            }
            delete pAGDoc;
        }
    }
    free (FileDlg.m_ofn.lpstrFile);
}


void CMainFrame::OnCardList()
{
    CString csFilters;

    csFilters = "Create and Print Files (*.ctp)";
    csFilters += "|";
    csFilters += "*.ctp";
    csFilters += "|";
    csFilters += "|";

    CFileDialog FileDlg (TRUE, NULL, NULL,
      OFN_HIDEREADONLY | OFN_FILEMUSTEXIST | OFN_ALLOWMULTISELECT,
      csFilters, AfxGetMainWnd ());
    FileDlg.m_ofn.lpstrTitle = "Select File(s) for Card List";
    FileDlg.m_ofn.lpstrFile = (char *) malloc (MAX_FILEBUF);
    *FileDlg.m_ofn.lpstrFile = 0;
    FileDlg.m_ofn.nMaxFile = MAX_FILEBUF;

    if (FileDlg.DoModal () == IDOK)
    {
        CProgressDlg ProgDlg (IDS_PROGRESS_CARDLIST);
        ProgDlg.Create();

        int nFileCount = 0;
        POSITION Pos = FileDlg.GetStartPosition ();
        while (Pos != NULL)
        {
            nFileCount++;
            FileDlg.GetNextPathName (Pos);
        }
        ProgDlg.SetRange (0, nFileCount);

        FILE *output = fopen ("cardlist.csv", "w");
        if (output == NULL)
        {
            MessageBox ("Can't open output file", NULL, MB_OK);
            return;
        }

        Pos = FileDlg.GetStartPosition ();
        while (Pos != NULL)
        {
            CString csFileName = FileDlg.GetNextPathName (Pos);
            CString csMsg = "Processing ";
            csMsg += csFileName;
            ProgDlg.SetStatus (csMsg);
            if (ProgDlg.CheckCancelButton ())
                break;

            CAGDoc *pAGDoc = new CAGDoc ();
            ifstream In (csFileName, ios::in | ios::nocreate | ios::binary);

            if (pAGDoc->Read (In))
            {
                if (ProgDlg.CheckCancelButton ())
                    break;

                char szFName[_MAX_FNAME];
                _splitpath (csFileName, NULL, NULL, szFName, NULL);
                szFName[7] = 0;

                AGDOCTYPE DocType = pAGDoc->GetDocType ();
                fprintf (output, "%s, %s, %s\n", szFName,
                  (char *) ((DocType == DOC_CARDHV || DocType == DOC_CARDHH) ? "s" : "q"),
                  (char *) ((DocType == DOC_CARDHV || DocType == DOC_CARDFV) ? "p" : "l"));

                ProgDlg.StepIt ();
                if (ProgDlg.CheckCancelButton ())
                    break;
            }
            delete pAGDoc;
        }

        fclose (output);
    }
    free (FileDlg.m_ofn.lpstrFile);
}

oid CMainFrame::OnPrintOptions()

    m_nPrintOption = GetCurrentMessage ()->wParam;

oid CMainFrame::OnUpdatePrintBoth(CCmdUI* pCmdUI)

    pCmdUI->SetRadio (m_nPrintOption == ID_PRINTBOTH);

oid CMainFrame::OnUpdatePrintDefault(CCmdUI* pCmdUI)

    pCmdUI->SetRadio (m_nPrintOption == ID_PRINTDEFAULT);
```

```
void CMainFrame::OnUpdatePrintQuarter(CCmdUI* pCmdUI)
{
    pCmdUI->SetRadio (m_nPrintOption == ID_PRINTQUARTER);
}

void CMainFrame::OnUpdatePrintSingle(CCmdUI* pCmdUI)
{
    pCmdUI->SetRadio (m_nPrintOption == ID_PRINTSINGLE);
}


void CMainFrame::CreateThumb (const CAGPage *pPage, int nMaxSize,
  const char *pszFileName)
{
    SIZE sizePage;
    pPage->GetPageSize (&sizePage);

    double xScale = (double) nMaxSize / (double) sizePage.cx;
    double yScale = (double) nMaxSize / (double) sizePage.cy;
    if (xScale < yScale)
        yScale = xScale;
    else
        xScale = yScale;

    BITMAPINFO  bi;
    memset (&bi, 0, sizeof (bi));
    bi.bmiHeader.biSize = sizeof (BITMAPINFOHEADER);
    bi.bmiHeader.biWidth = (int) ((sizePage.cx * xScale) + 0.5);
    bi.bmiHeader.biHeight = (int) ((sizePage.cy * yScale) + 0.5);
    bi.bmiHeader.biPlanes = 1;
    bi.bmiHeader.biBitCount = 24;
    bi.bmiHeader.biCompression = BI_RGB;
    bi.bmiHeader.biXPelsPerMeter = 3938;
    bi.bmiHeader.biYPelsPerMeter = 3938;

    BYTE *pBits = NULL;
    CAGDIBSectionDC dc (&bi, DIB_RGB_COLORS, &pBits);

    RECT DestRect = {0, 0, bi.bmiHeader.biWidth, bi.bmiHeader.biHeight};
    ::FillRect (dc.GetHDC (), &DestRect, (HBRUSH) ::GetStockObject (WHITE_BRUSH));

    CAGMatrix Matrix (xScale, 0, 0, yScale);
    dc.SetDeviceMatrix (Matrix);
    pPage->Draw (dc);


    BITMAPINFOHEADER biNew = bi.bmiHeader;
    biNew.biWidth /= 2;
    biNew.biHeight /= 2;
    biNew.biSizeImage = 0;

    BITMAPINFOHEADER *pdibDest = (BITMAPINFOHEADER *) GlobalAllocPtr (GHND, DibSize (&biNew));
    if (pdibDest)
    {
        *pdibDest = biNew;

        Image dst, src;
        src.xsize = (int) bi.bmiHeader.biWidth;
        src.ysize = (int) bi.bmiHeader.biHeight;
        src.data = pBits;
        src.span = DibWidthBytes (&bi.bmiHeader);
        dst.xsize = (int) pdibDest->biWidth;
        dst.ysize = (int) pdibDest->biHeight;
        dst.data = (BYTE *) DibPtr(pdibDest);
        dst.span = DibWidthBytes (pdibDest);

        zoom (&dst, &src);

        WriteBMP (pszFileName, (BITMAPINFO *)pdibDest, DibPtr (pdibDest));

        GlobalFreePtr (pdibDest);
    }
```

```
//  WriteBMP (pszFileName, &bi, pBits);
}

void CMainFrame::WriteBMP (const char *pszFileName, const BITMAPINFO *pbmi,
  const BYTE *pBits)
{
    int nImageSize = DibSizeImage (&pbmi->bmiHeader);

    BITMAPFILEHEADER bf;
    memset (&bf, 0, sizeof (bf));
    bf.bfType = 0x4d42; /* 'BM' */
    bf.bfSize = sizeof (bf) + pbmi->bmiHeader.biSize + nImageSize;
    bf.bfOffBits = sizeof (bf) + pbmi->bmiHeader.biSize;

    FILE *output = fopen (pszFileName, "wb");
    if (output)
    {
        fwrite (&bf, sizeof (bf), 1, output);
        fwrite (&pbmi->bmiHeader, pbmi->bmiHeader.biSize, 1, output);
        fwrite (pBits, nImageSize, 1, output);
        fclose (output);
    }
}


//=============================================================================//
// The following code is from Filtered Image Rescaling by Dale Schumacher   //
// in Graphics Gems III                                                      //
//=============================================================================//
#define M_PI            3.141592
#define WHITE_PIXEL     (255)
#define BLACK_PIXEL     (0)
#define CLAMP(v,l,h)    ((v)<(l) ? (l) : (v) > (h) ? (h) : v)

typedef struct {
    BYTE b;
    BYTE g;
    BYTE r;
} Pixel;

typedef struct {
    int pixel;
    double  weight;
} CONTRIB;

typedef struct {
    int n;      /* number of contributors */
    CONTRIB *p;     /* pointer to list of contributions */
} CLIST;

CLIST   *contrib;       /* array of contribution lists */

void get_row(Pixel *row, Image *image, int y)
{
    if((y < 0) || (y >= image->ysize)) {
        return;
    }
    hmemcpy(row,
        image->data + ((long) y * image->span),
        (sizeof(Pixel) * image->xsize));
}

void get_column(Pixel *column, Image *image, int x)

    int i, d;
    BYTE *p;

    if((x < 0) || (x >= image->xsize)) {
        return;
    }
    d = image->span;
    for(i = image->ysize, p = image->data + (x * sizeof (Pixel)); i-- > 0; p += d) {
        column->r = p[2];
        column->g = p[1];
        column->b = p[0];
        column++;
    }
}

void put_pixel(Image *image, int x, int y, Pixel *data)
{
    static Image *im = NULL;
    static int yy = -1;
    static BYTE *p = NULL;

    if((x < 0) || (x >= image->xsize) || (y < 0) || (y >= image->ysize)) {
        return;
    }
    if((im != image) || (yy != y)) {
        im = image;
        yy = y;
        p = image->data + ((long) y * image->span);
    }
    BYTE *pTemp = p + (x * sizeof (Pixel));
    *pTemp++ = data->b;
    *pTemp++ = data->g;
    *pTemp = data->r;
}

inline double sinc(double x)
{
    x *= M_PI;
    if(x != 0) return(sin(x) / x);
    return(1.0);
}

#define Lanczos3_support    (3.0)
inline double Lanczos3_filter(double t)
{
    if(t < 0) t = -t;
    if(t < 3.0) return(sinc(t) * sinc(t/3.0));
    return(0.0);
}

void zoom (Image *dst, Image *src)
{
    Image tmp;          /* intermediate image */
    double xscale, yscale;      /* zoom scale factors */
    int i, j, k;            /* loop variables */
    int n;              /* pixel number */
    double center, left, right; /* filter calculation variables */
    double width, fscale, weight;   /* filter calculation variables */
    Pixel *raster;          /* a row or column of pixels */

    /* create intermediate image to hold horizontal zoom */
    tmp.xsize = dst->xsize;
    tmp.ysize = src->ysize;
    tmp.data = (BYTE *) GlobalAllocPtr (GHND,
      (long) tmp.xsize * tmp.ysize * sizeof (Pixel));
    if (tmp.data == NULL)
        return;
    tmp.span = tmp.xsize * sizeof (Pixel);

    xscale = (double) dst->xsize / (double) src->xsize;
    yscale = (double) dst->ysize / (double) src->ysize;

    /* pre-calculate filter contributions for a row */
    contrib = (CLIST *)GlobalAllocPtr (GHND, dst->xsize * sizeof(CLIST));
    if(xscale < 1.0) {
        width = Lanczos3_support / xscale;
        fscale = 1.0 / xscale;
        for(i = 0; i < dst->xsize; ++i) {
            contrib[i].n = 0;
            contrib[i].p = (CONTRIB *)GlobalAllocPtr (GHND,
              (long) (width * 2 + 1) * sizeof(CONTRIB));
            center = (double) i / xscale;
            left = ceil(center - width);
            right = floor(center + width);
            for(j = (int) left; j <= (int) right; ++j) {
                weight = center - (double) j;
                weight = Lanczos3_filter(weight / fscale) / fscale;
                if(j < 0) {
                    n = -j;
                } else if(j >= src->xsize) {
                    n = (src->xsize - j) + src->xsize - 1;
                } else {
                    n = j;
                }
                k = contrib[i].n++;
                contrib[i].p[k].pixel = n;
                contrib[i].p[k].weight = weight;
            }
        }
    } else {
        for(i = 0; i < dst->xsize; ++i) {
            contrib[i].n = 0;
            contrib[i].p = (CONTRIB *)GlobalAllocPtr (GHND,
              (Lanczos3_support * 2 + 1) * sizeof(CONTRIB));
            center = (double) i / xscale;
            left = ceil(center - Lanczos3_support);
            right = floor(center + Lanczos3_support);
            for(j = (int) left; j <= (int) right; ++j) {
                weight = center - (double) j;
                weight = Lanczos3_filter(weight);
                if(j < 0) {
                    n = -j;
                } else if(j >= src->xsize) {
                    n = (src->xsize - j) + src->xsize - 1;
                } else {
                    n = j;
                }
                k = contrib[i].n++;
                contrib[i].p[k].pixel = n;
                contrib[i].p[k].weight = weight;
            }
        }
    }

    /* apply filter to zoom horizontally from src to tmp */
    raster = (Pixel *)GlobalAllocPtr (GHND, src->xsize * sizeof(Pixel));
    for(k = 0; k < tmp.ysize; ++k) {
        get_row(raster, src, k);
        for(i = 0; i < tmp.xsize; ++i) {
            double rweight = 0.0;
            double gweight = 0.0;
            double bweight = 0.0;
            for(j = 0; j < contrib[i].n; ++j) {
                rweight += raster[contrib[i].p[j].pixel].r
                    * contrib[i].p[j].weight;
                gweight += raster[contrib[i].p[j].pixel].g
                    * contrib[i].p[j].weight;
                bweight += raster[contrib[i].p[j].pixel].b
                    * contrib[i].p[j].weight;
            }
            Pixel p;
            p.r = (BYTE) CLAMP(rweight, BLACK_PIXEL, WHITE_PIXEL);
            p.g = (BYTE) CLAMP(gweight, BLACK_PIXEL, WHITE_PIXEL);
            p.b = (BYTE) CLAMP(bweight, BLACK_PIXEL, WHITE_PIXEL);
            // Since this output is destined for the printer, force to white
            if (p.r > 250 && p.g > 250 && p.b > 250)
                p.r = p.g = p.b = 255;
            put_pixel(&tmp, i, k, &p);
        }
    }
    GlobalFreePtr (raster);

    /* free the memory allocated for horizontal filter weights */
    for(i = 0; i < tmp.xsize; ++i) {
        GlobalFreePtr (contrib[i].p);
    }
    GlobalFreePtr (contrib);

    /* pre-calculate filter contributions for a column */
    contrib = (CLIST *)GlobalAllocPtr (GHND, dst->ysize * sizeof(CLIST));
    if(yscale < 1.0) {
        width = Lanczos3_support / yscale;
        fscale = 1.0 / yscale;
        for(i = 0; i < dst->ysize; ++i) {
            contrib[i].n = 0;
            contrib[i].p = (CONTRIB *)GlobalAllocPtr (GHND,
              (long) (width * 2 + 1) * sizeof(CONTRIB));
            center = (double) i / yscale;
            left = ceil(center - width);
            right = floor(center + width);
            for(j = (int) left; j <= (int) right; ++j) {
                weight = center - (double) j;
                weight = Lanczos3_filter(weight / fscale) / fscale;
                if(j < 0) {
                    n = -j;
                } else if(j >= tmp.ysize) {
                    n = (tmp.ysize - j) + tmp.ysize - 1;
                } else {
                    n = j;
                }
                k = contrib[i].n++;
                contrib[i].p[k].pixel = n;
                contrib[i].p[k].weight = weight;
            }
        }
    } else {
        for(i = 0; i < dst->ysize; ++i) {
            contrib[i].n = 0;
            contrib[i].p = (CONTRIB *)GlobalAllocPtr (GHND,
              (Lanczos3_support * 2 + 1) * sizeof(CONTRIB));
            center = (double) i / yscale;
            left = ceil(center - Lanczos3_support);
            right = floor(center + Lanczos3_support);
            for(j = (int) left; j <= (int) right; ++j) {
                weight = center - (double) j;
                weight = Lanczos3_filter(weight);
                if(j < 0) {
                    n = -j;
                } else if(j >= tmp.ysize) {
                    n = (tmp.ysize - j) + tmp.ysize - 1;
                } else {
                    n = j;
                }
                k = contrib[i].n++;
                contrib[i].p[k].pixel = n;
                contrib[i].p[k].weight = weight;
            }
        }
    }

    /* apply filter to zoom vertically from tmp to dst */
    raster = (Pixel *)GlobalAllocPtr (GHND, tmp.ysize * sizeof(Pixel));
    for(k = 0; k < dst->xsize; ++k) {
        get_column(raster, &tmp, k);
        for(i = 0; i < dst->ysize; ++i) {
            double rweight = 0.0;
            double gweight = 0.0;
            double bweight = 0.0;
            for(j = 0; j < contrib[i].n; ++j) {
                rweight += raster[contrib[i].p[j].pixel].r
                    * contrib[i].p[j].weight;
                gweight += raster[contrib[i].p[j].pixel].g
                    * contrib[i].p[j].weight;
                bweight += raster[contrib[i].p[j].pixel].b
                    * contrib[i].p[j].weight;
            }
            Pixel p;
            p.r = (BYTE) CLAMP(rweight, BLACK_PIXEL, WHITE_PIXEL);
            p.g = (BYTE) CLAMP(gweight, BLACK_PIXEL, WHITE_PIXEL);
            p.b = (BYTE) CLAMP(bweight, BLACK_PIXEL, WHITE_PIXEL);
            // Since this output is destined for the printer, force to white
            if (p.r > 250 && p.g > 250 && p.b > 250)
                p.r = p.g = p.b = 255;
            put_pixel(dst, k, i, &p);
        }
    }
    GlobalFreePtr (raster);

    /* free the memory allocated for vertical filter weights */
    for(i = 0; i < dst->ysize; ++i) {
        GlobalFreePtr (contrib[i].p);
    }
    GlobalFreePtr (contrib);

    GlobalFreePtr (tmp.data);
}
```

```
// MainFrm.h : interface of the CMainFrame class
//
/////////////////////////////////////////////////////////////////////////////

#if !defined(AFX_MAINFRM_H__64AE5209_3509_11D3_9331_0080C6F796A1__INCLUDED_)
#define AFX_MAINFRM_H__64AE5209_3509_11D3_9331_0080C6F796A1__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

#include "AGPage.h"

class CMainFrame : public CFrameWnd
{
public:
	CMainFrame();

// Attributes
public:

// Operations
public:

// Overrides
	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CMainFrame)
	virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
	//}}AFX_VIRTUAL

// Implementation
public:
	virtual ~CMainFrame();
#ifdef _DEBUG
	virtual void AssertValid() const;
	virtual void Dump(CDumpContext& dc) const;
#endif

protected:
	void CreateThumb (const CAGPage *pPage, int nMaxSize,
	  const char *pszFileName);
	void WriteBMP (const char *pszFileName, const BITMAPINFO *pbmi,
	  const BYTE *pBits);

// Generated message map functions
protected:
	//{{AFX_MSG(CMainFrame)
	afx_msg void OnBatchPrint();
	afx_msg void OnBuildThumbs();
	afx_msg void OnCardList();
	afx_msg void OnPrintOptions();
	afx_msg void OnUpdatePrintBoth(CCmdUI* pCmdUI);
	afx_msg void OnUpdatePrintDefault(CCmdUI* pCmdUI);
	afx_msg void OnUpdatePrintQuarter(CCmdUI* pCmdUI);
	afx_msg void OnUpdatePrintSingle(CCmdUI* pCmdUI);
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()

protected:
	int m_nPrintOption;
};

/////////////////////////////////////////////////////////////////////////////

//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately before the previous li
ne.

#endif // !defined(AFX_MAINFRM_H__64AE5209_3509_11D3_9331_0080C6F796A1__INCLUDED_)
```

```
//  ProgDlg.cpp : implementation file
// CG: This file was added by the Progress Dialog component

#include "stdafx.h"
#include "resource.h"
#include "ProgDlg.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif


/////////////////////////////////////////////////////////////////////////////
// CProgressDlg dialog
//
CProgressDlg::CProgressDlg(UINT nCaptionID)
{
    m_nCaptionID = CG_IDS_PROGRESS_CAPTION;
    if (nCaptionID != 0)
        m_nCaptionID = nCaptionID;

    m_bCancel=FALSE;
    m_nLower=0;
    m_nUpper=100;
    m_nStep=1;
    //{{AFX_DATA_INIT(CProgressDlg)
    // NOTE: the ClassWizard will add member initialization here
    //}}AFX_DATA_INIT
    m_bParentDisabled = FALSE;
}

CProgressDlg::~CProgressDlg()
{
    if(m_hWnd!=NULL)
      DestroyWindow();
}

BOOL CProgressDlg::DestroyWindow()
{
    ReEnableParent();
    return CDialog::DestroyWindow();
}

void CProgressDlg::ReEnableParent()
{
    if(m_bParentDisabled && (m_pParentWnd!=NULL))
      m_pParentWnd->EnableWindow(TRUE);
    m_bParentDisabled=FALSE;
}


BOOL CProgressDlg::Create(CWnd *pParent)
{
    // Get the true parent of the dialog
    m_pParentWnd = CWnd::GetSafeOwner(pParent);

    // m_bParentDisabled is used to re-enable the parent window
    // when the dialog is destroyed. So we don't want to set
    // it to TRUE unless the parent was already enabled.

    if ((m_pParentWnd!=NULL) && m_pParentWnd->IsWindowEnabled())
    {
        m_pParentWnd->EnableWindow(FALSE);
        m_bParentDisabled = TRUE;
    }

    if (!CDialog::Create(CProgressDlg::IDD,pParent))
    {
        ReEnableParent();
        return FALSE;
    }
    return TRUE;
}


void CProgressDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CProgressDlg)
    DDX_Control(pDX, CG_IDC_PROGDLG_PROGRESS, m_Progress);
    //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CProgressDlg, CDialog)
    //{{AFX_MSG_MAP(CProgressDlg)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()


void CProgressDlg::SetStatus(LPCTSTR lpszMessage)
{
    ASSERT(m_hWnd); // Don't call this _before_ the dialog has
                    // been created. Can be called from OnInitDialog
    CWnd *pWndStatus = GetDlgItem(CG_IDC_PROGDLG_STATUS);

    // Verify that the static text control exists
    ASSERT(pWndStatus!=NULL);
    pWndStatus->SetWindowText(lpszMessage);
}


void CProgressDlg::OnCancel()
{
    m_bCancel=TRUE;
}


void CProgressDlg::SetRange(int nLower,int nUpper)
{
    m_nLower = nLower;
    m_nUpper = nUpper;
    m_Progress.SetRange(nLower,nUpper);
}


int CProgressDlg::SetPos(int nPos)
{
    PumpMessages();
    int iResult = m_Progress.SetPos(nPos);
    UpdatePercent(nPos);
    return iResult;
}


int CProgressDlg::SetStep(int nStep)
{
    m_nStep = nStep;                    // Store for later use in calculating percentage
    return m_Progress.SetStep(nStep);
}


int CProgressDlg::OffsetPos(int nPos)
{
    PumpMessages();
    int iResult = m_Progress.OffsetPos(nPos);
    UpdatePercent(iResult+nPos);
    return iResult;
}


int CProgressDlg::StepIt()
{
    PumpMessages();
    int iResult = m_Progress.StepIt();
    UpdatePercent(iResult+m_nStep);
    return iResult;
}


void CProgressDlg::PumpMessages()
{
    // Must call Create() before using the dialog
    ASSERT(m_hWnd!=NULL);

    MSG msg;
    // Handle dialog messages
    while(PeekMessage(&msg, NULL, 0, 0, PM_REMOVE))
    {
      if(!IsDialogMessage(&msg))
      {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
      }
    }
}


BOOL CProgressDlg::CheckCancelButton()
{
    // Process all pending messages
    PumpMessages();

    // Reset m_bCancel to FALSE so that
    // CheckCancelButton returns FALSE until the user
    // clicks Cancel again. This will allow you to call
    // CheckCancelButton and still continue the operation.
    // If m_bCancel stayed TRUE, then the next call to
    // CheckCancelButton would always return TRUE

    BOOL bResult = m_bCancel;
    m_bCancel = FALSE;

    return bResult;
}


void CProgressDlg::UpdatePercent(int nNewPos)
{
    CWnd *pWndPercent = GetDlgItem(CG_IDC_PROGDLG_PERCENT);
    int nPercent;

    int nDivisor = m_nUpper - m_nLower;
    ASSERT(nDivisor > 0);               // m_nLower should be smaller than m_nUpper

    int nDividend = (nNewPos - m_nLower);
    ASSERT(nDividend >= 0);             // Current position should be greater than m_nLower

    nPercent = nDividend * 100 / nDivisor;

    // Since the Progress Control wraps, we will wrap the percentage
    // along with it. However, don't reset 100% back to 0%
    if(nPercent!=100)
      nPercent %= 100;

    // Display the percentage
    CString strBuf;
    strBuf.Format(_T("%d%c"),nPercent,_T('%'));

    CString strCur;                     // get current percentage
    pWndPercent->GetWindowText(strCur);

    if (strCur != strBuf)
        pWndPercent->SetWindowText(strBuf);
}
```

```
/////////////////////////////////////////////////////////////////////////////
// OnInitDialog
//
BOOL CProgressDlg::OnInitDialog()
{
    CDialog::OnInitDialog();
    m_Progress.SetRange(m_nLower,m_nUpper);
    m_Progress.SetStep(m_nStep);
    m_Progress.SetPos(m_nLower);

    CString strCaption;
    VERIFY(strCaption.LoadString(m_nCaptionID));
    SetWindowText(strCaption);

    return TRUE;
}
```

```
// ProgDlg.h : header file
// CG: This file was added by the Progress Dialog component

/////////////////////////////////////////////////////////////////////////////
// CProgressDlg dialog

#ifndef __PROGDLG_H__
#define __PROGDLG_H__

class CProgressDlg : public CDialog
{
// Construction / Destruction
public:
    CProgressDlg(UINT nCaptionID = 0);   // standard constructor
    ~CProgressDlg();

    BOOL Create(CWnd *pParent=NULL);

    // Checking for Cancel button
    BOOL CheckCancelButton();
    // Progress Dialog manipulation
    void SetStatus(LPCTSTR lpszMessage);
    void SetRange(int nLower,int nUpper);
    int  SetStep(int nStep);
    int  SetPos(int nPos);
    int  OffsetPos(int nPos);
    int  StepIt();

// Dialog Data
    //{{AFX_DATA(CProgressDlg)
    enum { IDD = CG_IDD_PROGRESS };
    CProgressCtrl   m_Progress;
    //}}AFX_DATA

// Overrides
    // ClassWizard generated virtual function overrides
    //{{AFX_VIRTUAL(CProgressDlg)
    public:
    virtual BOOL DestroyWindow();
    protected:
    virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
    //}}AFX_VIRTUAL

// Implementation
protected:
    UINT m_nCaptionID;
    int m_nLower;
    int m_nUpper;
    int m_nStep;

    BOOL m_bCancel;
    BOOL m_bParentDisabled;

    void ReEnableParent();

    virtual void OnCancel();
    virtual void OnOK() {};
    void UpdatePercent(int nCurrent);
    void PumpMessages();

    // Generated message map functions
    //{{AFX_MSG(CProgressDlg)
    virtual BOOL OnInitDialog();
    //}}AFX_MSG
    DECLARE_MESSAGE_MAP()
};

#endif // __PROGDLG_H__
```

```
//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by CTPUtil.rc
//
#define IDD_ABOUTBOX                    100
#define CG_IDD_PROGRESS                 102
#define CG_IDS_PROGRESS_CAPTION         103
#define IDS_PROGRESS_THUMBS             104
#define IDS_PROGRESS_CARDLIST           105
#define IDR_MAINFRAME                   128
#define IDR_CTPUTITYPE                  129
#define IDC_VERSION                     1000
#define CG_IDC_PROGDLG_PROGRESS         1003
#define CG_IDC_PROGDLG_PERCENT          1004
#define CG_IDC_PROGDLG_STATUS           1005
#define ID_BUILDTHUMBS                  32771
#define ID_BATCHPRINT                   32772
#define ID_PRINTDEFAULT                 32773
#define ID_PRINTSINGLE                  32774
#define ID_PRINTQUARTER                 32775
#define ID_PRINTBOTH                    32776
#define ID_CARDLIST                     32777

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_3D_CONTROLS                     1
#define _APS_NEXT_RESOURCE_VALUE        130
#define _APS_NEXT_COMMAND_VALUE         32778
#define _APS_NEXT_CONTROL_VALUE         1002
#define _APS_NEXT_SYMED_VALUE           104
#endif
#endif
```

[illegible]

File: Work\CrtPrt\CTPUtil\StdAfx.cpp   Page: 1

```
// stdafx.cpp : source file that includes just the standard includes
//  CTPUtil.pch will be the pre-compiled header
//  stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"
```

```
// stdafx.h : include file for standard system include files,
//  or project specific include files that are used frequently, but
//      are changed infrequently
//

#if !defined(AFX_STDAFX_H__64AE5207_3509_11D3_9331_0080C6F796A1__INCLUDED_)
#define AFX_STDAFX_H__64AE5207_3509_11D3_9331_0080C6F796A1__INCLUDED_

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

#define VC_EXTRALEAN        // Exclude rarely-used stuff from Windows headers

#include <afxwin.h>         // MFC core and standard components
#include <afxext.h>         // MFC extensions
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>         // MFC support for Windows Common Controls
#endif // _AFX_NO_AFXCMN_SUPPORT


//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately before the previous li
ne.

#include <vector>

#define WIDTH(r)        ((r).right - (r).left)
#define HEIGHT(r)       ((r).bottom - (r).top)
#define SWAP(a,b)       ((a) ^= (b),(b) ^= (a),(a) ^= (b))
#define APP_RESOLUTION  1440

#endif // !defined(AFX_STDAFX_H__64AE5207_3509_11D3_9331_0080C6F796A1__INCLUDED_)
```

```
# Microsoft Developer Studio Project File - Name="CTPInst" - Package Owner=<4>
# Microsoft Developer Studio Generated Build File, Format Version 5.00
# ** DO NOT EDIT **

# TARGTYPE "Win32 (x86) Application" 0x0101

CFG=CTPInst - Win32 Debug
!MESSAGE This is not a valid makefile. To build this project using NMAKE,
!MESSAGE use the Export Makefile command and run
!MESSAGE
!MESSAGE NMAKE /f "CTPInst.mak".
!MESSAGE
!MESSAGE You can specify a configuration when running NMAKE
!MESSAGE by defining the macro CFG on the command line. For example:
!MESSAGE
!MESSAGE NMAKE /f "CTPInst.mak" CFG="CTPInst - Win32 Debug"
!MESSAGE
!MESSAGE Possible choices for configuration are:
!MESSAGE
!MESSAGE "CTPInst - Win32 Release" (based on "Win32 (x86) Application")
!MESSAGE "CTPInst - Win32 Debug" (based on "Win32 (x86) Application")
!MESSAGE

# Begin Project
# PROP Scc_ProjName ""$/CTPInst", MFAAAAAA"
# PROP Scc_LocalPath "."
CPP=cl.exe
MTL=midl.exe
RSC=rc.exe

!IF  "$(CFG)" == "CTPInst - Win32 Release"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "Release"
# PROP BASE Intermediate_Dir "Release"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Release"
# PROP Intermediate_Dir "Release"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_WINDOWS" /YX /FD /c
# ADD CPP /nologo /W3 /GX /O1 /D "WIN32" /D "NDEBUG" /D "_WINDOWS" /FD /c
# SUBTRACT CPP /YX
# ADD BASE MTL /nologo /D "NDEBUG" /mktyplib203 /o NUL /win32
# ADD MTL /nologo /D "NDEBUG" /mktyplib203 /o NUL /win32
# ADD BASE RSC /l 0x409 /d "NDEBUG"
# ADD RSC /l 0x409 /d "NDEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.lib shell32.l
ib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo /subsystem:windows /machine:I386
# ADD LINK32 kernel32.lib user32.lib gdi32.lib advapi32.lib version.lib /nologo /subsystem:windows /
machine:I386 /nodefaultlib
# SUBTRACT LINK32 /map

!ELSEIF  "$(CFG)" == "CTPInst - Win32 Debug"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "Debug"
# PROP BASE Intermediate_Dir "Debug"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Debug"
# PROP Intermediate_Dir "Debug"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /W3 /Gm /GX /Zi /Od /D "WIN32" /D "_DEBUG" /D "_WINDOWS" /YX /FD /c
# ADD CPP /nologo /W3 /Gm /GX /Zi /Od /D "WIN32" /D "_DEBUG" /D "_WINDOWS" /FD /c
# SUBTRACT CPP /YX
# ADD BASE MTL /nologo /D "_DEBUG" /mktyplib203 /o NUL /win32
# ADD MTL /nologo /D "_DEBUG" /mktyplib203 /o NUL /win32
# ADD BASE RSC /l 0x409 /d "_DEBUG"
# ADD RSC /l 0x409 /d "_DEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib advapi32.lib shell32.l
ib ole32.lib oleaut32.lib uuid.lib odbc32.lib odbccp32.lib /nologo /subsystem:windows /debug /machin
e:I386 /pdbtype:sept
# ADD LINK32 kernel32.lib user32.lib gdi32.lib advapi32.lib version.lib /nologo /subsystem:windows /
debug /machine:I386 /nodefaultlib /pdbtype:sept

!ENDIF

# Begin Target

# Name "CTPInst - Win32 Release"
# Name "CTPInst - Win32 Debug"
# Begin Source File

SOURCE=.\CTPInst.cpp
# End Source File
# Begin Source File

SOURCE=.\CTPInst.rc

!IF  "$(CFG)" == "CTPInst - Win32 Release"

!ELSEIF  "$(CFG)" == "CTPInst - Win32 Debug"

!ENDIF

# End Source File
# Begin Source File

SOURCE=.\Npctp.dl_
# End Source File
# Begin Source File

SOURCE=..\NpCtp\Release\NPCTP.dll

!IF  "$(CFG)" == "CTPInst - Win32 Release"

# Begin Custom Build
InputPath=..\NpCtp\Release\NPCTP.dll

"NpCtp.dl_" : $(SOURCE) "$(INTDIR)" "$(OUTDIR)"
    compress $(InputPath) NpCtp.dl_

# End Custom Build

!ELSEIF  "$(CFG)" == "CTPInst - Win32 Debug"

# Begin Custom Build
InputPath=..\NpCtp\Release\NPCTP.dll

"NpCtp.dl_" : $(SOURCE) "$(INTDIR)" "$(OUTDIR)"
    compress $(InputPath) NpCtp.dl_

# End Custom Build

!ENDIF

# End Source File
# Begin Source File

SOURCE=.\resource.h
# End Source File
# Begin Source File

SOURCE=.\version.h
# End Source File
# End Target
# End Project
```

```
#include <windows.h>
#include "resource.h"

#define MSGBOX_TITLE    "Installer"
#define SRCFILE         "npctp.dl_"
#define DESTFILE        "NpCtp.dll"


//
// Eliminate the need for linking in the CRT libraries
//
extern "C" void WinMainCRTStartup()
{
    //
    // Get the Navigator plugin directory
    //
    char szInstallDir[_MAX_PATH] = "";

    HKEY hKey;
    if (RegOpenKeyEx(HKEY_CURRENT_USER, "Software\\Netscape\\Netscape Navigator\\Main",
        0, KEY_READ, &hKey) == ERROR_SUCCESS)
    {
        DWORD dwSize = sizeof(szInstallDir);
        RegQueryValueEx(hKey, "Install Directory", NULL, NULL, (BYTE *)szInstallDir, &dwSize);
        RegCloseKey(hKey);
    }

    if (!szInstallDir[0])
    {
        MessageBox(NULL, "Netscape Navigator 3 installation directory registry entry not found.",
                    MSGBOX_TITLE, MB_OK);
        ExitProcess(0);
    }
    else
    {
        if (szInstallDir[lstrlen(szInstallDir) - 1] != '\\')
            lstrcat(szInstallDir, "\\");
        lstrcat(szInstallDir, "program\\plugins");
    }


    //
    // Tell the user that we're going to install the plug-in
    //
    char szMsg[512];
    wsprintf(szMsg, "This is the American Greetings Create and Print Plug-in installer\n"
                "for Netscape Navigator version 3\n\n"
                "It will install the plug-in into -\n%s\n\n"
                "Press OK to install.",
                szInstallDir);

    if (MessageBox(NULL, szMsg, MSGBOX_TITLE, MB_OKCANCEL) == IDOK)
    {
        //
        // Copy the compressed plug-in to the temp directory
        //
        char szSrcDir[_MAX_PATH];
        GetTempPath(sizeof(szSrcDir), szSrcDir);

        char szCompressedFile[_MAX_PATH];
        lstrcpy(szCompressedFile, szSrcDir);
        if (szCompressedFile[lstrlen(szCompressedFile) - 1] != '\\')
            lstrcat(szCompressedFile, "\\");
        lstrcat(szCompressedFile, SRCFILE);

        OFSTRUCT of;
        HANDLE hFile = (HANDLE)OpenFile(szCompressedFile, &of, OF_CREATE | OF_WRITE);
        if (hFile == (HANDLE)HFILE_ERROR)
        {
            MessageBox(NULL, "Error extracting plug-in to temp directory.",
                        MSGBOX_TITLE, MB_OK);
            ExitProcess(0);
        }

        HRSRC hResInfo = FindResource(NULL, MAKEINTRESOURCE(IDR_COMPRESSED), "COMPRESSED");
        HGLOBAL hResData = LoadResource(NULL, hResInfo);
        DWORD dwSize = SizeofResource(NULL, hResInfo);
        void *pData = LockResource(hResData);
        DWORD dwBytesWritten;
        WriteFile(hFile, pData, dwSize, &dwBytesWritten, NULL);
        CloseHandle(hFile);

        //
        // Decompress the plug-in to the Netscape plug-in directory
        //
        char szTempFile[_MAX_PATH] = "";
        UINT uTmpFileLen = 0;
        DWORD dwResult = 0;

        dwResult = VerInstallFile(0, SRCFILE, DESTFILE, szSrcDir, szInstallDir, "",
                                  szTempFile, &uTmpFileLen);

        if (dwResult == 0)
            MessageBox(NULL, "The plug-in was successfully installed.", MSGBOX_TITLE, MB_OK);
        else if (dwResult & VIF_SRCOLD)
        {
            MessageBox(NULL, "You already have a newer version of the plug-in installed.",
                       MSGBOX_TITLE, MB_OK);
        }
        else if (dwResult & VIF_FILEINUSE)
        {
            MessageBox(NULL, "You already have a version of the plug-in installed and it is currentl
y in use.\n"
                       "You will have to exit Netscape before installing the new version",
                       MSGBOX_TITLE, MB_OK);
        }
        else if (dwResult & VIF_OUTOFSPACE)
        {
            MessageBox(NULL, "You do not have enough disk space to install the plug-in.",
                       MSGBOX_TITLE, MB_OK);
        }
        else if ((dwResult & VIF_WRITEPROT) || (dwResult & VIF_TEMPFILE))
        {
            dwResult = VerInstallFile(VIFF_FORCEINSTALL, SRCFILE, DESTFILE, szSrcDir,
                                      szInstallDir, "", szTempFile, &uTmpFileLen);
            if (dwResult & VIF_WRITEPROT)
            {
                MessageBox(NULL,
                           "You already have a version of the plug-in installed and it is write pro
tected.\n"
                           "You will have to remove the read only attribute before installing the n
ew version.",
                           MSGBOX_TITLE, MB_OK);
            }
            else if (dwResult)
            {
                char szMsg[50];
                wsprintf(szMsg, "Installation failed.  Error %x", dwResult);
                MessageBox(NULL, szMsg, MSGBOX_TITLE, MB_OK);
            }
            else
                MessageBox(NULL, "The plug-in was successfully installed.", MSGBOX_TITLE, MB_OK);
        }
        else
        {
            char szMsg[50];
            wsprintf(szMsg, "Installation failed.  Error %x", dwResult);
            MessageBox(NULL, szMsg, MSGBOX_TITLE, MB_OK);
        }

        //
        // Clean up the temporary files
        //
        DeleteFile(szCompressedFile);
        if (szTempFile[0])
```

```
            DeleteFile(szTempFile);
    }
    else
        MessageBox(NULL, "Installation aborted.", MSGBOX_TITLE, MB_OK);

    ExitProcess(0);
}
```

```
# Microsoft Developer Studio Project File - Name="CTPInst" - Packa
ge Owner=<4>
# Microsoft Developer Studio Generated Build File, Format Version
6.00
# ** DO NOT EDIT **

# TARGTYPE "Win32 (x86) Application" 0x0101

CFG=CTPInst - Win32 Debug
!MESSAGE This is not a valid makefile. To build this project using
 NMAKE,
!MESSAGE use the Export Makefile command and run
!MESSAGE
!MESSAGE NMAKE /f "CTPInst.mak".
!MESSAGE
!MESSAGE You can specify a configuration when running NMAKE
!MESSAGE by defining the macro CFG on the command line. For exampl
e:
!MESSAGE
!MESSAGE NMAKE /f "CTPInst.mak" CFG="CTPInst - Win32 Debug"
!MESSAGE
!MESSAGE Possible choices for configuration are:
!MESSAGE
!MESSAGE "CTPInst - Win32 Release" (based on "Win32 (x86) Applicat
ion")
!MESSAGE "CTPInst - Win32 Debug" (based on "Win32 (x86) Applicatio
n")
!MESSAGE

# Begin Project
# PROP AllowPerConfigDependencies 0
# PROP Scc_ProjName ""$/CTPInst", MFAAAAAA"
# PROP Scc_LocalPath "."
CPP=cl.exe
MTL=midl.exe
RSC=rc.exe

!IF  "$(CFG)" == "CTPInst - Win32 Release"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "Release"
# PROP BASE Intermediate_Dir "Release"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Release"
# PROP Intermediate_Dir "Release"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_WIN
DOWS" /YX /FD /c
# ADD CPP /nologo /W3 /GX /O1 /D "WIN32" /D "NDEBUG" /D "_WINDOWS"
 /FD /c
# SUBTRACT CPP /YX
# ADD BASE MTL /nologo /D "NDEBUG" /mktyplib203 /o "NUL" /win32
# ADD MTL /nologo /D "NDEBUG" /mktyplib203 /o "NUL" /win32
# ADD BASE RSC /l 0x409 /d "NDEBUG"
# ADD RSC /l 0x409 /d "NDEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib c
omdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.l
ib odbc32.lib odbccp32.lib /nologo /subsystem:windows /machine:I38
6
# ADD LINK32 kernel32.lib user32.lib gdi32.lib advapi32.lib versio
n.lib /nologo /subsystem:windows /machine:I386 /nodefaultlib
# SUBTRACT LINK32 /map

!ELSEIF  "$(CFG)" == "CTPInst - Win32 Debug"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "Debug"
# PROP BASE Intermediate_Dir "Debug"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Debug"
# PROP Intermediate_Dir "Debug"
# PROP Ignore_Export_Lib 0
# PROP Target_Dir ""
# ADD BASE CPP /nologo /W3 /Gm /GX /Zi /Od /D "WIN32" /D "_DEBUG"
/D "_WINDOWS" /YX /FD /c
# ADD CPP /nologo /W3 /Gm /GX /ZI /Od /D "WIN32" /D "_DEBUG" /D "_
WINDOWS" /FD /c
# SUBTRACT CPP /YX
# ADD BASE MTL /nologo /D "_DEBUG" /mktyplib203 /o "NUL" /win32
# ADD MTL /nologo /D "_DEBUG" /mktyplib203 /o "NUL" /win32
# ADD BASE RSC /l 0x409 /d "_DEBUG"
# ADD RSC /l 0x409 /d "_DEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib c
omdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.l
ib odbc32.lib odbccp32.lib /nologo /subsystem:windows /debug /mach
ine:I386 /pdbtype:sept
# ADD LINK32 kernel32.lib user32.lib gdi32.lib advapi32.lib versio
n.lib /nologo /subsystem:windows /debug /machine:I386 /nodefaultli
b /pdbtype:sept

!ENDIF

# Begin Target

# Name "CTPInst - Win32 Release"
# Name "CTPInst - Win32 Debug"
# Begin Source File

SOURCE=.\CTPInst.cpp
# End Source File
# Begin Source File

SOURCE=.\CTPInst.rc
# End Source File
# Begin Source File

SOURCE=.\Npctp.dl_
# End Source File
# Begin Source File

SOURCE=..\NpCtp\Release\NPCTP.dll

!IF  "$(CFG)" == "CTPInst - Win32 Release"

# Begin Custom Build
InputPath=..\NpCtp\Release\NPCTP.dll

"NpCtp.dl_" : $(SOURCE) "$(INTDIR)" "$(OUTDIR)"
	compress $(InputPath) NpCtp.dl_

# End Custom Build

!ELSEIF  "$(CFG)" == "CTPInst - Win32 Debug"
```

```
# Begin Custom Build
InputPath=..\NpCtp\Release\NPCTP.dll

"NpCtp.dl_" : $(SOURCE) "$(INTDIR)" "$(OUTDIR)"
	compress $(InputPath) NpCtp.dl_

# End Custom Build

!ENDIF

# End Source File
# Begin Source File

SOURCE=.\resource.h
# End Source File
# Begin Source File

SOURCE=.\version.h
# End Source File
# End Target
# End Project
```

```
Microsoft Developer Studio Workspace File, Format Version 6.00
# WARNING: DO NOT EDIT OR DELETE THIS WORKSPACE FILE!

######################################################################
#############

Project: "CTPInst"=.\CTPInst.dsp - Package Owner=<4>

Package=<5>
{{{
    begin source code control
    "$/CTPInst", MFAAAAAA
    .
    end source code control
}}}

Package=<4>
{{{
}}}

######################################################################
#############

Global:

Package=<5>
{{{
    begin source code control
    "$/CTPInst", MFAAAAAA
    .
    end source code control
}}}

Package=<3>
{{{
}}}

######################################################################
#############
```

```
<html>
<body>
<pre>
<h1>Build Log</h1>
<h3>
--------------------Configuration: CTPInst - Win32 Debug--------------------
</h3>
<h3>Command Lines</h3>
Creating temporary file "c:\windows\TEMP\RSP7184.BAT" with contents
[
@echo off
compress ..\NpCtp\Release\NPCTP.dll NpCtp.dl_
]
Creating command line "c:\windows\TEMP\RSP7184.BAT"
Creating command line "rc.exe /l 0x409 /fo"Debug/CTPInst.res" /d "_DEBUG" "C:\Work\CrtPrt\CTPInst\CT
PInst.rc""
Performing Custom Build Step on ..\NpCtp\Release\NPCTP.dll
Bad command or file name
Creating temporary file "c:\windows\TEMP\RSP7195.TMP" with contents
[
/nologo /MLd /W3 /Gm /GX /ZI /Od /D "WIN32" /D "_DEBUG" /D "_WINDOWS" /Fo"Debug/" /Fd"Debug/" /FD /c

"C:\Work\CrtPrt\CTPInst\CTPInst.cpp"
]
Creating command line "cl.exe @c:\windows\TEMP\RSP7195.TMP"
Creating command line "link.exe kernel32.lib user32.lib gdi32.lib advapi32.lib version.lib /nologo /
subsystem:windows /incremental:yes /pdb:"Debug/CTPInst.pdb" /debug /machine:I386 /nodefaultlib /out:
"Debug/CTPInst.exe" /pdbtype:sept  .\Debug\CTPInst.obj .\Debug\CTPInst.res "
<h3>Output Window</h3>
Compiling resources...

C:\Work\CrtPrt\CTPInst\CTPInst.rc (99): error RC2135 : file not found: Npctp.dl_
Error executing rc.exe.


<h3>Results</h3>
CTPInst.exe - 1 error(s), 0 warning(s)
</pre>
</body>
</html>
```

```
//Microsoft Developer Studio generated resource script.
//
#include "resource.h"

#define APSTUDIO_READONLY_SYMBOLS
//////////////////////////////////////////////////////////////////////
///////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"
#include "version.h"

//////////////////////////////////////////////////////////////////////
///////////
#undef APSTUDIO_READONLY_SYMBOLS

//////////////////////////////////////////////////////////////////////
///////////
// English (U.S.) resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
#pragma code_page(1252)
#endif //_WIN32

#ifndef _MAC
//////////////////////////////////////////////////////////////////////
///////////
//
// Version
//

VS_VERSION_INFO VERSIONINFO
 FILEVERSION VER_FILE_VERSION
 PRODUCTVERSION VER_PRODUCT_VERSION
 FILEFLAGSMASK 0x3fL
#ifdef _DEBUG
 FILEFLAGS 0x1L
#else
 FILEFLAGS 0x0L
#endif
 FILEOS 0x40004L
 FILETYPE 0x1L
 FILESUBTYPE 0x0L
```

```
BEGIN
    BLOCK "StringFileInfo"
    BEGIN
        BLOCK "040904b0"
        BEGIN
            VALUE "CompanyName", VER_COMPANY
            VALUE "FileDescription", "Create and Print Plug-in ins
taller\0"
            VALUE "FileVersion", VER_FILE_VERSION_STR
            VALUE "InternalName", "CTPInst\0"
            VALUE "LegalCopyright", VER_COPYRIGHT
            VALUE "OriginalFilename", "CTPInst.exe\0"
            VALUE "ProductName", "Create and Print\0"
            VALUE "ProductVersion", VER_PRODUCT_VERSION_STR
        END
    END
    BLOCK "VarFileInfo"
    BEGIN
        VALUE "Translation", 0x409, 1200
    END
END

#endif    // !_MAC


#ifdef APSTUDIO_INVOKED
///////////////////////////////////////////////////////////////////
///////////
//
// TEXTINCLUDE
//

1 TEXTINCLUDE DISCARDABLE
BEGIN
    "resource.h\0"
END

2 TEXTINCLUDE DISCARDABLE
BEGIN
    "#include ""afxres.h""\r\n"
    "#include ""version.h""\r\n"
    "\0"
END

3 TEXTINCLUDE DISCARDABLE
BEGIN
```

```
    "\r\n"
    "\0"
END

#endif    // APSTUDIO_INVOKED


//////////////////////////////////////////////////////////////////
///////////
//
// COMPRESSED
//

IDR_COMPRESSED          COMPRESSED DISCARDABLE  "Npctp.dl_"
#endif    // English (U.S.) resources
//////////////////////////////////////////////////////////////////
///////////



#ifndef APSTUDIO_INVOKED
//////////////////////////////////////////////////////////////////
///////////
//
// Generated from the TEXTINCLUDE 3 resource.
//


//////////////////////////////////////////////////////////////////
///////////
#endif    // not APSTUDIO_INVOKED
```

```
//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by CTPInst.rc
//
#define IDR_COMPRESSED                  103

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NO_MFC                     1
#define _APS_NEXT_RESOURCE_VALUE        104
#define _APS_NEXT_COMMAND_VALUE         40001
#define _APS_NEXT_CONTROL_VALUE         1000
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

```
cabarc -s 6144 n AxCtp.cab AxCtp.dll AxCtp.inf
\inetsdk\bin\signcode -spc mycredentials.spc -v agkey.pvk -t http://timestamp.verisign.com/scripts/t
imstamp.dll axctp.cab
```

[illegible]

```
[Version]
Signature="$Chicago$"
AdvancedINF=2.0

[Add.Code]
AxCtp.dll=AxCtp.dll

[AxCtp.dll]
file-win32-x86=thiscab
clsid={38578BF0-0ABB-11D3-9330-0080C6F796A1}
FileVersion=1,0,0,0
RegesterServer=yes
```

```
[Version]
Signature="$Chicago$"
AdvancedINF=2.0

[Add.Code]
AxCtp.dll=AxCtp.dll

[AxCtp.dll]
file-win32-x86=thiscab
clsid={38578BF0-0ABB-11D3-9330-0080C6F796A1}
FileVersion=1,0,0,0
RegesterServer=yes
```

```
HKCR
{
    Ctp.Ctp.1 = s 'Ctp Class'
    {
        CLSID = s '{38578BF0-0ABB-11D3-9330-0080C6F796A1}'
    }
    Ctp.Ctp = s 'Ctp Class'
    {
        CLSID = s '{38578BF0-0ABB-11D3-9330-0080C6F796A1}'
        CurVer = s 'Ctp.Ctp.1'
    }
    NoRemove CLSID
    {
        ForceRemove {38578BF0-0ABB-11D3-9330-0080C6F796A1} = s 'Ctp Class'
        {
            ProgID = s 'Ctp.Ctp.1'
            VersionIndependentProgID = s 'Ctp.Ctp'
            ForceRemove 'Programmable'
            InprocServer32 = s '%MODULE%'
            {
                val ThreadingModel = s 'Apartment'
            }
            ForceRemove 'Control'
            ForceRemove 'Programmable'
            ForceRemove 'Insertable'
            ForceRemove 'ToolboxBitmap32' = s '%MODULE%, 1'
            'MiscStatus' = s '0'
            {
                '1' = s '131473'
            }
            'TypeLib' = s '{38578BF1-0ABB-11D3-9330-0080C6F796A1}'
            'Version' = s '1.0'
        }
    }
}
```

```
// Conditional alert.
function cAlert (message)
{
    if (!this.silent)
        alert(message);
}

// Variable indicating whether or not installation should proceed.
var bInstall = true;

// Make sure Java is enabled before doing anything else.
if (! navigator.javaEnabled())
{
    bInstall = false;
    cAlert ("Java must be enabled to install.");
}

// Make sure installation is attempted on correct machine architecture.
else if (navigator.platform != "Win32")
{
    bInstall = false;
    cAlert ("This plug-in only runs on Win32 platforms.");
}

// If all conditions look good, proceed with the installation.
if (bInstall)
{
    // Create a version object and a software update object
    var vi = new netscape.softupdate.VersionInfo(1, 0, 0, 0);
    var su = new netscape.softupdate.SoftwareUpdate(this, "American Greetings Create and Print Plug-
in");

    // Initialize bAbort.
    var bAbort = false;

    // Start the install process
    var err = su.StartInstall("plugins/AmericanGreetings/", vi,
        netscape.softupdate.SoftwareUpdate.FULL_INSTALL);

    if (err != 0)
        bAbort = true;
    else
    {
        // Find the plug-ins directory on the user's machine
        PIFolder = su.GetFolder("Plugins");

        // Install the files. Unpack them and list where they go
        err = su.AddSubcomponent("CTP DLL", vi, "NpCtp.dll",
            PIFolder, "", this.force);

        bAbort = bAbort || (err != 0);
    }

    // Unless there was a problem, move files to final location
    // and update the Client Version Registry
    if (bAbort)
    {
        cAlert ("Installation error. Aborting.");
        su.AbortInstall();
    }
    else
    {
        err = su.FinalizeInstall();

        // Refresh list of available plug-ins
        if (err == 0)
            navigator.plugins.refresh(true);
        else if (err == 999)
            cAlert("You must reboot to finish this installation.");
        else
            cAlert ("Installation error. Aborting.");
    }
```

```
//
// Don't forget to update the version number in Ctp.rgs, AxCtp.inf, NpCtp.js
//

#define VER_COMPANY                 "American Greetings.com\0"
#define VER_COPYRIGHT               "Copyright © 1999 American Greetings.com\0"

#define VER_PRODUCT_VERSION         1,0,0,0
#define VER_PRODUCT_VERSION_STR     "1.0"
#define VER_FILE_VERSION            1,0,0,0
#define VER_FILE_VERSION_STR        "1.0"
```

```
deltree /y src
del NpCtp.jar
md src
copy NpCtp.js src\NpCtp.js
copy ..\Npctp\Release\NpCtp.dll src\NpCtp.dll
signtool\signtool -d"D:\Program Files\Netscape\Users\default" -c9 -i NpCtp.js -k "American Greetings
's VeriSign, Inc. ID" -Z NpCtp.jar src
```

```
// Conditional alert.
function cAlert (message)
{
    if (!this.silent)
        alert(message);
}

// Variable indicating whether or not installation should proceed.
var bInstall = true;

// Make sure Java is enabled before doing anything else.
if (! navigator.javaEnabled())
{
    bInstall = false;
    cAlert ("Java must be enabled to install.");
}

// Make sure installation is attempted on correct machine architecture.
else if (navigator.platform != "Win32")
{
    bInstall = false;
    cAlert ("This plug-in only runs on Win32 platforms.");
}

// If all conditions look good, proceed with the installation.
if (bInstall)
{
    // Create a version object and a software update object
    var vi = new netscape.softupdate.VersionInfo(1, 0, 0, 0);
    var su = new netscape.softupdate.SoftwareUpdate(this, "American Greetings Create and Print Plug-
in");

    // Initialize bAbort.
    var bAbort = false;

    // Start the install process
    var err = su.StartInstall("plugins/AmericanGreetings/", vi,
        netscape.softupdate.SoftwareUpdate.FULL_INSTALL);

    if (err != 0)
        bAbort = true;
    else
    {
        // Find the plug-ins directory on the user's machine
        PIFolder = su.GetFolder("Plugins");

        // Install the files. Unpack them and list where they go
        err = su.AddSubcomponent("CTP DLL", vi, "NpCtp.dll",
            PIFolder, "", this.force);

        bAbort = bAbort || (err != 0);
    }

    // Unless there was a problem, move files to final location
    // and update the Client Version Registry
    if (bAbort)
    {
        cAlert ("Installation error. Aborting.");
        su.AbortInstall();
    }
    else
    {
        err = su.FinalizeInstall();

        // Refresh list of available plug-ins
        if (err == 0)
            navigator.plugins.refresh(true);
        else if (err == 999)
            cAlert("You must reboot to finish this installation.");
        else
            cAlert ("Installation error. Aborting.");
    }
}
```

```
Microsoft Developer Studio Workspace File, Format Version 5.00
# WARNING: DO NOT EDIT OR DELETE THIS WORKSPACE FILE!

###################################################################
#############

Project: "Stonehnd"=.\Stonehnd.dsp - Package Owner=<4>

Package=<5>
{{{
    begin source code control
    "$/Stonehnd", CGAAAAAA
    .
    end source code control
}}}

Package=<4>
{{{
}}}

###################################################################
#############

Global:

Package=<5>
{{{
    begin source code control
    "$/Stonehnd", CGAAAAAA
    .
    end source code control
}}}

Package=<3>
{{{
}}}

###################################################################
#############
```

```
SOURCE=.\Scstcach.h
# End Source File
# Begin Source File

SOURCE=.\Scstream.h
# End Source File
# Begin Source File

SOURCE=.\Scstyle.h
# End Source File
# Begin Source File

SOURCE=.\Sctbobj.h
# End Source File
# Begin Source File

SOURCE=.\Sctextli.h
# End Source File
# Begin Source File

SOURCE=.\Sctypes.h
# End Source File
# Begin Source File

SOURCE=.\Ufont.h
# End Source File
# Begin Source File

SOURCE=.\Univstr.h
# End Source File
# End Group
# End Target
# End Project
# End Source File
# Begin Source File

SOURCE=.\Scparagr.h
# End Source File
# Begin Source File

SOURCE=.\Scpolygo.h
# End Source File
# Begin Source File

SOURCE=.\Scpubobj.h
# End Source File
# Begin Source File

SOURCE=.\Scrange.h
# End Source File
# Begin Source File

SOURCE=.\Screfdat.h
# End Source File
# Begin Source File

SOURCE=.\Scregion.h
# End Source File
# Begin Source File

SOURCE=.\scrubi.h
# End Source File
# Begin Source File

SOURCE=.\Scselect.h
# End Source File
# Begin Source File

SOURCE=.\Scset.h
# End Source File
# Begin Source File

SOURCE=.\Scsetjmp.h
# End Source File
# Begin Source File

SOURCE=.\Scspcrec.h
# End Source File
# Begin Source File
```

```
SOURCE=.\Sccharex.h
# End Source File
# Begin Source File

SOURCE=.\Scchflag.h
# End Source File
# Begin Source File

SOURCE=.\Sccolumn.h
# End Source File
# Begin Source File

SOURCE=.\Scctype.h
# End Source File
# Begin Source File

SOURCE=.\Scdbcsdt.h
# End Source File
# Begin Source File

SOURCE=.\Scexcept.h
# End Source File
# Begin Source File

SOURCE=.\Scfileio.h
# End Source File
# Begin Source File

SOURCE=.\Scglobda.h
# End Source File
# Begin Source File

SOURCE=.\Scmacint.h
# End Source File
# Begin Source File

SOURCE=.\Scmem.h
# End Source File
# Begin Source File

SOURCE=.\Scmemarr.h
# End Source File
# Begin Source File

SOURCE=.\Scobject.h
```

```
# End Source File
# Begin Source File

SOURCE=.\Univstr.cpp
# End Source File
# End Group
# Begin Group "Header Files"

# PROP Default_Filter "h"
# Begin Source File

SOURCE=.\Refcnt.h
# End Source File
# Begin Source File

SOURCE=.\Scannota.h
# End Source File
# Begin Source File

SOURCE=.\Scappint.h
# End Source File
# Begin Source File

SOURCE=.\Scapptex.h
# End Source File
# Begin Source File

SOURCE=.\Scapptyp.h
# End Source File
# Begin Source File

SOURCE=.\Scarray.h
# End Source File
# Begin Source File

SOURCE=.\Scbezier.h
# End Source File
# Begin Source File

SOURCE=.\Scbreak.h
# End Source File
# Begin Source File

SOURCE=.\Sccallbk.h
# End Source File
# Begin Source File
```

```
SOURCE=.\Scselect.cpp
# End Source File
# Begin Source File

SOURCE=.\Scset.cpp
# End Source File
# Begin Source File

SOURCE=.\Scspcrec.cpp
# End Source File
# Begin Source File

SOURCE=.\Scstcach.cpp
# End Source File
# Begin Source File

SOURCE=.\Scstiter.cpp
# End Source File
# Begin Source File

SOURCE=.\Scstream.cpp
# End Source File
# Begin Source File

SOURCE=.\Scstyle.cpp
# End Source File
# Begin Source File

SOURCE=.\Sctbobj.cpp
# End Source File
# Begin Source File

SOURCE=.\Sctextch.cpp
# End Source File
# Begin Source File

SOURCE=.\Sctextli.cpp
# End Source File
# Begin Source File

SOURCE=.\Sctxtlnm.cpp
# End Source File
# Begin Source File

SOURCE=.\Ufont.cpp
```

```
# End Source File
# Begin Source File

SOURCE=.\Schrect.cpp
# End Source File
# Begin Source File

SOURCE=.\Scmemarr.cpp
# End Source File
# Begin Source File

SOURCE=.\Scnshmem.cpp
# End Source File
# Begin Source File

SOURCE=.\Scobject.cpp
# End Source File
# Begin Source File

SOURCE=.\Scparag2.cpp
# End Source File
# Begin Source File

SOURCE=.\Scparag3.cpp
# End Source File
# Begin Source File

SOURCE=.\Scparagr.cpp
# End Source File
# Begin Source File

SOURCE=.\Scpolygo.cpp
# End Source File
# Begin Source File

SOURCE=.\Scregion.cpp
# End Source File
# Begin Source File

SOURCE=.\scrubi.cpp
# End Source File
# Begin Source File

SOURCE=.\Scselec2.cpp
# End Source File
# Begin Source File
```

```
SOURCE=.\Sccolinf.cpp
# End Source File
# Begin Source File

SOURCE=.\Sccolum2.cpp
# End Source File
# Begin Source File

SOURCE=.\Sccolum3.cpp
# End Source File
# Begin Source File

SOURCE=.\Sccolumn.cpp
# End Source File
# Begin Source File

SOURCE=.\Sccspecl.cpp
# End Source File
# Begin Source File

SOURCE=.\Scctype.cpp
# End Source File
# Begin Source File

SOURCE=.\Scdbcsdt.cpp
# End Source File
# Begin Source File

SOURCE=.\Scdebug.cpp
# End Source File
# Begin Source File

SOURCE=.\Scdeftmp.cpp
# End Source File
# Begin Source File

SOURCE=.\Scexcept.cpp
# End Source File
# Begin Source File

SOURCE=.\Scfileio.cpp
# End Source File
# Begin Source File

SOURCE=.\Scglobda.cpp
```

```
# Name "Stonehnd - Win32 Release"
# Name "Stonehnd - Win32 Debug"
# Begin Group "Source Files"

# PROP Default_Filter "cpp"
# Begin Source File

SOURCE=.\Sc_chmap.cpp
# End Source File
# Begin Source File

SOURCE=.\Sc_spchg.cpp
# End Source File
# Begin Source File

SOURCE=.\Sc_sysco.cpp
# End Source File
# Begin Source File

SOURCE=.\Sc_utlwi.cpp
# End Source File
# Begin Source File

SOURCE=.\Scapi.cpp
# End Source File
# Begin Source File

SOURCE=.\Scapptex.cpp
# End Source File
# Begin Source File

SOURCE=.\Scarray.cpp
# End Source File
# Begin Source File

SOURCE=.\Scbezble.cpp
# End Source File
# Begin Source File

SOURCE=.\Scbezier.cpp
# End Source File
# Begin Source File

SOURCE=.\Scbreak.cpp
# End Source File
# Begin Source File
# PROP Intermediate_Dir "Release"
# PROP Target_Dir ""
# ADD BASE CPP /nologo /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_WIN
DOWS" /YX /FD /c
# ADD CPP /nologo /Zp2 /MT /W3 /GX /O1 /D "WIN32" /D "NDEBUG" /D "
_WINDOWS" /FD /c
# SUBTRACT CPP /YX /Yc /Yu
# ADD BASE RSC /l 0x409
# ADD RSC /l 0x409
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LIB32=link.exe -lib
# ADD BASE LIB32 /nologo
# ADD LIB32 /nologo

!ELSEIF  "$(CFG)" == "Stonehnd - Win32 Debug"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "Debug"
# PROP BASE Intermediate_Dir "Debug"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Debug"
# PROP Intermediate_Dir "Debug"
# PROP Target_Dir ""
# ADD BASE CPP /nologo /W3 /GX /Z7 /Od /D "WIN32" /D "_DEBUG" /D "
_WINDOWS" /YX /FD /c
# ADD CPP /nologo /Zp2 /MTd /W3 /GX /Z7 /Od /D "WIN32" /D "_DEBUG"
 /D "_WINDOWS" /FD /c
# SUBTRACT CPP /YX
# ADD BASE RSC /l 0x409
# ADD RSC /l 0x409
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
LIB32=link.exe -lib
# ADD BASE LIB32 /nologo
# ADD LIB32 /nologo

!ENDIF

# Begin Target
# Microsoft Developer Studio Project File - Name="Stonehnd" - Pack
age Owner=<4>
# Microsoft Developer Studio Generated Build File, Format Version
6.00
# ** DO NOT EDIT **

# TARGTYPE "Win32 (x86) Static Library" 0x0104

CFG=Stonehnd - Win32 Debug
!MESSAGE This is not a valid makefile. To build this project using
 NMAKE,
!MESSAGE use the Export Makefile command and run
!MESSAGE
!MESSAGE NMAKE /f "Stonehnd.mak".
!MESSAGE
!MESSAGE You can specify a configuration when running NMAKE
!MESSAGE by defining the macro CFG on the command line. For exampl
e:
!MESSAGE
!MESSAGE NMAKE /f "Stonehnd.mak" CFG="Stonehnd - Win32 Debug"
!MESSAGE
!MESSAGE Possible choices for configuration are:
!MESSAGE
!MESSAGE "Stonehnd - Win32 Release" (based on "Win32 (x86) Static
Library")
!MESSAGE "Stonehnd - Win32 Debug" (based on "Win32 (x86) Static Li
brary")
!MESSAGE

# Begin Project
# PROP AllowPerConfigDependencies 0
# PROP Scc_ProjName ""$/Stonehnd", CGAAAAAA"
# PROP Scc_LocalPath "."
CPP=cl.exe
RSC=rc.exe

!IF  "$(CFG)" == "Stonehnd - Win32 Release"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "Release"
# PROP BASE Intermediate_Dir "Release"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Release"
```

```
                len = 0;
                stUnivChar* p = new stUnivChar[ strlen( str ) ];
                ptr = (const stUnivChar*)p;
                len = strlen( str );
                const char* ch = str;
                for ( int i = 0; *ch; )
                    p[i++] = *ch++;
                return *this;
            }

    void    reverse();

    int operator==( const stUnivString& );
    int operator!=( const stUnivString& );
};

void u2char( void* buf, const stUnivString& ustr );

#endif
```

```
#ifndef UNIVSTR_H_
#define UNIVSTR_H_

#include <string.h>

// a string class that can hold ascii, jis or ISO10646
typedef unsigned short    stUnivChar;
struct stUnivString {
  const stUnivChar *ptr;
  unsigned long len;
};

class UniversalString : public stUnivString {
public:
    UniversalString()
            {
                ptr = 0;
                len = 0;
            }
    UniversalString( const char* str )
            {
                stUnivChar* p = new stUnivChar[ strlen( str ) ];
                ptr = (const stUnivChar*)p;
                len = strlen( str );
                const char* ch = str;
                for ( int i = 0; *ch; )
                    p[i++] = *ch++;

            }
    UniversalString( const stUnivString& stustr )
            {
                stUnivChar* p = new stUnivChar[ stustr.len ];
                ptr = (const stUnivChar*)p;
                len = stustr.len;
                for ( unsigned i = 0; i < stustr.len; i++ )
                    p[i] = stustr.ptr[i];
            }
    ~UniversalString()
            {
                delete [] (stUnivChar*)ptr;
            }
    UniversalString& operator=( const stUnivString& stustr )
            {
                delete [] (stUnivChar*)ptr;
                stUnivChar* p;
                if ( stustr.len )
                    p = new stUnivChar[ stustr.len ];
                else
                    p = 0;
                ptr = (const stUnivChar*)p;
                len = stustr.len;
                for ( unsigned i = 0; i < stustr.len; i++ )
                    p[i] = stustr.ptr[i];
                return *this;
            }
    UniversalString& operator=( const UniversalString& stustr )
            {
                delete [] (stUnivChar*)ptr;
                stUnivChar* p;
                if ( stustr.len )
                    p = new stUnivChar[ stustr.len ];
                else
                    p = 0;
                ptr = (const stUnivChar*)p;
                len = stustr.len;
                for ( unsigned i = 0; i < stustr.len; i++ )
                    p[i] = stustr.ptr[i];
                return *this;
            }

    UniversalString& operator=( const char* str )
            {
                delete [] (stUnivChar*)ptr, ptr = 0;
```

```
                        fHasDamage( false ),
                        fHasRepaint( false ),
                        fImmediateRedisplay( false ) {}

PUBLISHED:
    scColumn*           fColumnID;          // @member fColumnID | Stonehand name.
    APPColumn           fAPPName;           // @member fAPPName | An <t APPColumn>
                                            // holding the client's name.

        // the recorded width and depth in the container
    MicroPoint          fWidth;             // @member fWidth | Current container width.
    MicroPoint          fDepth;             // @member fDepth | Current container depth.
    scXRect             fExRect;            // @member fExRect | An <c scXRect> that contains
                                            // the ink extents of the container.
    Bool                fAdditionalText;    // @member fAdditionalText | True
                                            // is text flows out the bottom of
                                            // this container.

    Bool                fHasDamage;
    scXRect             fDamageRect;         /* damaged area of column */

    Bool                fHasRepaint;
    scXRect             fRepaintRect;        /* area to repaint */

    Bool                fImmediateRedisplay;    // @member fImmediateRedisplay |
                                                // True if there is a need
                                                // immediate redisplay info.
    scImmediateRedisp   fImmediateArea;         // @member fImmediateArea | Area
                                                // to redisplay immediately.
};


/* ==================================================================== */

class scCOLRefData;

// @class This class contains an array of structures that hold information
// about redisplay. For each effected column there will be a corresponding
// class <c scColRedisplay> with information about what areas ( in local coordinates ) need
// to be redispalyed.

class scRedispList : public scMemArray {
PUBLISHED:
    status          CL_GetColumnData( APPColumn, scColRedisplay& ) const;

public:
                    scRedispList( ) :
                        scMemArray( sizeof( scColRedisplay ) ){}

    void            ReInit( void );
    void            AddColumn( const scCOLRefData& colRefData );
    void            AddColumn( scColumn*, scXRect& );

    void            SetImmediateRect( scColumn*, const scImmediateRedisp& );

protected:
    scColRedisplay* FindCell( const scColumn* ) const;
    void            AddCell( scColumn* );
};


/* ==================================================================== */
/* ==================================================================== */
/* ==================================================================== */


#endif
```

```
/*================== COLUMN LEVEL MESSAGES ==================*/
/* a note on functions returning a pointer to ColRect's,
   many of these operations may have impact on multiple containers,
   therefore a pointer to a ColRect is a null terminated chain
   of ColRect's, with each ColRect containing the scColumn* & appName,
   its new set of extents and its damaged area, a typical thing to do
   when one receives one of these might be something like this:

    for ( ; colrect->columnID != NULL; colrect++ ) {
        ColumnLocalToGlobal( colrect->appName, &colrect->exRect );
        UpdateColumnExtents( colrect->appName, &colrect->exRect );
        ColumnLocalToGlobal( colrect->appName, &colrect->damageRect );
        InvalHRect( &colrect->damageRect );
    }

 */


/* FLAG LINE */
typedef enum eFlagTypes{
    okSet,
    overSet,            /* the line is overset */
    minExSet,           /* the minimum wordspace value is less than desired */
    maxExSet            /* the maximum wordspace value is exceeded */
} eFlagType;

struct scFlaggedLine {
    scXRect     lineExtents;            /* extents of line to flag */
    /* the following represent character characteristics of the
     * first character on the line
     */
    MicroPoint  xHite;
    MicroPoint  capHite;
    MicroPoint  ascHite;
    MicroPoint  desHite;
    MicroPoint  baseline;
    eFlagType   flag;                   /* type of flag */
                                /* used to flag lines */

// @class scImmediateRedisp is for immediate redisplay of text that has
// been altered in editting. This is useful for updateing a minimal number of
// lines during editting and letting the rest of the text get repaired
// on an update event. It is basically redisplaying only those lines
// which the cursor has been on, hitting a carriage return should force two
// lines to be redisplayed immediately, same with a backspace at the
// beggining of a line, normally only one line will be redisplayed
// IF THE OPERATION CROSSES COLUMNS WE WILL ONLY UPDATE THE COLUMN
// IN WHICH THE CURSOR ENDS UP IN.
// @xref <f SCCOL_UpdateLine>

class scImmediateRedisp {
public:
                scImmediateRedisp() :
                    fStartLine( 0 ),
                    fStopLine( 0 ) { }

    short       fStartLine;         // @cmember Start drawing this line.
    short       fStopLine;          // @cmember End drawing this line.
    scXRect     fImmediateRect;     // @cmember Erase this area.
};

// @class This contains the information needed to correctly redisplay
// a column after reformatting.

class scColRedisplay {
public:
                    scColRedisplay(){}
                    scColRedisplay( scColumn* col, APPColumn appcol ) :
                        fColumnID( col ),
                        fAPPName( appcol ),
                        fWidth( 0 ),
                        fDepth( 0 ),
                        fAdditionalText( false ),
    void        SetStream( scStream* stream )
                    {
                        fStream = stream;
                    }
    scStream*   GetStream( void ) const
                    {
                        return fStream;
                    }

public:
                scSpecLocList( scStream* stream ) :
                        fStream( stream ){}


            // append a paragraph terminator
    void        TermParagraph( int32 paraoffset, int32 offset )
                {
                    scSpecLocation chsploc;
                    scParaOffset& poffset   = chsploc.offset();
                    poffset.fParaOffset     = paraoffset;
                    poffset.fCharOffset     = offset;

                    Append( chsploc );
                }


#if SCDEBUG > 1
    void        DbgPrint( void ) const;
#endif

private:
    scStream*   fStream;        // @member fStream | Stream to which this list belongs.

};


/* ================================================================== */
/* ================ LINE INFO ======================================== */
/* ================================================================== */

class scLineInfoList : public scMemArray {
public:
            scLineInfoList() :
                    scMemArray( sizeof( scLineInfo ) ) {}
};


/* ================================================================== */

/* ================================================================== */
/* ==============   COLUMN REDISPLAY/DAMAGE REPORT  ================== */
/* ================================================================== */

/* a column id struct that Stonehand returns on many calls, used by APP to
 * determine damage extents for redisplay, the redisplay requires two areas
 * for correct redisplay, the damaged area - is the area that needs to
 * be repainted to patch the screen, the repaint area is the area that needs
 * to be repainted to update the screen, in most instances the union of these
 * two areas would be sufficient but if the user's interaction, reorigined
 * the column - an action that Stonehand
 * has no way of knowing, the control program may repair the screen with the
 * damaged area and then use the repaint area to update the screen.
 * all of the areas/rectangles are expressed in high precision rectangles and
 * are relative to the columns origin, the extents and repaint area
 * are in relation to the current origin, the damaged area is in relation
 * to the previous origin, no account has been made for any matrix
 * transformations.
 * The flagLineHandle is a handle to an array containing information about
 * lines that exceed nominal values (e.g. overset, word-space values
 * exceeded, etc ), it may be used to flag the lines visually for the user
 */
```

```
/* ====================   scSpecLocList   ========================= */
/* ================================================================= */
/* ================================================================= */
// The Char Spec list - used to retreive and set the spec associated
// with a particular character(s)

// @struct scParaOffset | is used to specify a postion in a stream.
class scParaOffset {
public:
            scParaOffset() :
                    fParaOffset( -1 ),
                    fCharOffset( -1 ){}

    int32   fParaOffset;            // @field Paragraph offset within stream.
    int32   fCharOffset;            // @field Character offset within paragraph.
};

// @struct scSpecLocation | is used to specify a position in the stream
// at which to start the application of a <t TypeSpec>.
// @xref <c scSpecLocList>
class scSpecLocation {
public:
                    scSpecLocation(){}
                    scSpecLocation( int32 poffset, int32 offset )
                    {
                        offset_.fParaOffset = poffset;
                        offset_.fCharOffset = offset;
                    }
                    scSpecLocation( int32     poffset,
                                   int32     offset,
                                   TypeSpec& spec )
                    {
                        offset_.fParaOffset = poffset;
                        offset_.fCharOffset = offset;
                        spec_ = spec;
                    }
    scParaOffset&   offset()
                    {
                        return offset_;
                    }
    const scParaOffset& offset() const
                    {
                        return offset_;
                    }

    TypeSpec&       spec()
                    {
                        return spec_;
                    }

    const TypeSpec& spec() const
                    {
                        return spec_;
                    }

private:
    scParaOffset    offset_;    // @field The <t scParaOffset> is
                                    // the stream location.
    TypeSpec        spec_;          // @field The <t TypeSpec> to be applied at
                                    // this stream location.
};


// @class The scSpecLocList is an array of <t scSpecLocation> that contain
// the stream position and the spec associated with each spec transition.
// @xref <c scTypeSpecList>
//

class scSpecLocList : public scSizeableArrayD< scSpecLocation > {
PUBLISHED:
    TypeSpec    GetFirstValidSpec( void ) const;
    TypeSpec    GetNthValidSpec( int ) const;
    TypeSpec    GetLastValidSpec( void ) const;
```

```
/*==================================================================

    File:       scpubobj

    $Header: /Projects/Toolbox/ct/SCPUBOBJ.H 2     5/30/97 8:45a Wmanis $

    Contains:   This contains some of the more specialized public
                data types that are NOT in sctypes.h.

    Written by: Manis

    Copyright:  (c) 1988-94 by Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

    @doc
==================================================================*/

#ifndef _H_SCPUBOBJ
#define _H_SCPUBOBJ

#include "scmemarr.h"
#include "scarray.h"

/* ==================================================================== */
/* ERROR HANDLING
/* ==================================================================== */

#ifdef API_RERAISE

    #define IGNORE_RERAISE      catch ( status stat_ ) {\
                                    throw( stat_ );\
                                }\
                                catch ( ... ) {\
                                    throw( scERRgeneral );\
                                }

#else

    #define IGNORE_RERAISE      catch ( status stat_ ) {\
                                    stat = stat_;\
                                }\
                                catch ( ... ) {\
                                    stat = scERRgeneral;\
                                }

#endif

/* ==================================================================== */
/* ==================================================================== */
/* ==================================================================== */
/* =====================   SCTYPESPECLIST   ========================== */
/* ==================================================================== */
/* ==================================================================== */
// @class The TypeSpec List used for retreiving lists of type specs.
// @xref <c scSpecLocList>

class scTypeSpecList : public scSizeableArrayD< TypeSpec > {
PUBLISHED:
    TypeSpec    CL_GetNth( int );

public:
    void    Insert( TypeSpec& );
};

/* ==================================================================== */
/* ==================================================================== */
```

```
                    else
                        return Inclusive_Sect( range );
                }

private:
    const long      fStart;
    const long      fEnd;
};

#endif /* _H_SCRANGE */
```

```
/*==================================================================

    File:       crange.h

    $Header: /Projects/Toolbox/ct/SCRANGE.H 2     5/30/97 8:45a Wmanis $

    Contains:   Tests ranges.

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

==================================================================*/

#ifndef _H_SCRANGE
#define _H_SCRANGE

#include "sctypes.h"

typedef enum eRangeSects {
    eExclusive,
    eInclusive,
    eStartInclusive,
    eEndInclusive
} eRangeSect;

class scRange {
public:
            scRange( long start, long end ) :
                fStart( start ),
                fEnd( end ) {}

    Bool    Exclusive_Sect( const scRange& range ) const
                {   return fStart < range.fEnd && fEnd > range.fStart; }

    Bool    Inclusive_Sect( const scRange& range ) const
                { return fStart <= range.fEnd && fEnd >= range.fStart; }

    Bool    InclusiveStart_Sect( const scRange& range ) const
                { return fStart <= range.fEnd && fEnd > range.fStart; }

    Bool    InclusiveEnd_Sect( const scRange& range ) const
                { return fStart < range.fEnd && fEnd >= range.fStart; }

    Bool    Inclusive( long val ) const
                {   return val >= fStart && val <= fEnd; }

    Bool    Exclusive( long val ) const
                {   return val > fStart && val < fEnd; }

    Bool    StartInclusive( long val ) const
                {   return val >= fStart && val < fEnd; }

    Bool    EndInclusive( long val ) const
                {   return val > fStart && val <= fEnd; }

    Bool    Sect( const scRange& range, eRangeSect eSect )
                {
                    if ( eSect == eExclusive )
                        return Exclusive_Sect( range );
                    else if ( eSect == eStartInclusive )
                        return InclusiveStart_Sect( range );
                    else if ( eSect == eEndInclusive )
                        return InclusiveEnd_Sect( range );
    void            SetFirstlinePos( MicroPoint mp )    { fFirstlinePos = mp; }
    MicroPoint      GetFirstlinePos( void ) const       { return fFirstlinePos; }

    void            SetRegion( HRgn* rgn )              { fRgn = rgn; }
    HRgn*           GetRegion( void ) const             { return fRgn; }

    scPARARefData   fPData;             // the current para ref data

    scColumn*       fCol;               // the column currently being reformatted
    scXRect         fLineDamage;        // lines marked as damaged
    TypeSpec        fFirstSpec;         // the first spec active in the column
    MicroPoint      fFirstlinePos;      // the position of the first line in the col
    scMuPoint       fPrevEnd;           // ending position of previous line
    scMuPoint       fSavedPrevEnd;      // end pos - saved
    HRgn*           fRgn;               // the active region
    HRgnHandle      fRgnH;

    scRedisplayStoredLine   fSavedLineState;
    scRedispList*           fRedispList;
};

/* ================================================================== */

#endif /* _H_SCREFDAT */
```

```
                // total number of lines formatted in para
    short           fLineNumber;

                // current number of lines hyphenated in a row
    short           fLinesHyphed;


                    // backing store and glyph array stuff
    CharRecordP     fCharRecs;

    scSpecRecord*   fStartSpecRec;
    scSpecRecord*   fCurSpecRec;
    long            fSpecCount;

#ifdef _RUBI_SUPPORT
    scRubiArray*    fAnnotations;
#endif

    // somewhere along in here we would add the unpositioned glyphs
    // and the positioned glyphs hooks


private:
    scColumn*       column_;

};


/* ==================================================================== */


class scCOLRefData {
public:
                    scCOLRefData( scRedispList* redisp ) :
                            fCol( 0 ),
                            fFirstSpec( 0 ),
                            fFirstlinePos( 0 ),
                            fRgn( 0 ),
                            fRedispList( redisp ) {}

    Bool            COLInit( scColumn*, scContUnit* );
    void            COLFini( Bool );

                    // free lines marks as invalid and collect their damaged area
    void            FreeInvalidLines( void );

        // this stores the existing lines and collects and repainting
        // of the lines necessary
    void            SaveLineList( void );

    void            PARAInit( scContUnit*, int, int, PrevParaData& );

    Bool            AllocGeometry( void );
    Bool            AllocLine( Bool leadRetry );

                    // delete excess lines in the paragraph we are currently formatting
    void            PARADeleteExcessLines( void );

    Bool            FindNextCol( DCState& );

                    // move from one column to the next resetting
                    // the appropriate values
    void            Transition( void );

    Bool            ResetOrphan( Bool testGetStrip );

    void            SetActive( scColumn* col );
    scColumn*       GetActive( void ) const         { return fCol; }

    void            SetFirstSpec( TypeSpec spec )       { fFirstSpec = spec; }
    TypeSpec        GetFirstSpec( void ) const          { return fFirstSpec; }
    void            SetPrevSpec( TypeSpec ps )          { fPrevSpec = ps; }
    TypeSpec        GetPrevSpec( void )                 { return fPrevSpec; }

    void            SetOrigin( const scMuPoint& org )   { fOrigin = org; }
    scMuPoint&      GetOrigin( void )                   { return fOrigin; }

    void            SetPrevLead( const scLEADRefData& cl )  { fPrevLead = cl; }
    scLEADRefData&  GetPrevLead( void )                 { return fPrevLead; }

    void            SetPara( scContUnit* p )               { fPara = p; }
    scContUnit*     GetPara( void )                     { return fPara; }

    void            SetBreakParams( const scParaColBreak&   pb )
                        {
                            fBreakParams = pb;
                        }
    scParaColBreak& GetBreakParams( void )
                        {
                            return fBreakParams;
                        }

    void            SetLinesHyphed( short lh )          { fLinesHyphed = lh; }
    short           GetLinesHyphed( void )              { return fLinesHyphed; }

    void            SetStartChar( CharRecordP ch  )     { fCharRecs = ch; }
    CharRecordP     GetStartChar( void )                { return fCharRecs; }

    void            SetStartSpecRecord( scSpecRecord* spr ) { fStartSpecRec = spr; }
    scSpecRecord*   GetStartSpecRecord( void )             { return fStartSpecRec; }

    void            SetSpecCount( long count )          { fSpecCount = count; }
    long            GetSpecCount( void )                { return fSpecCount; }

    scSpecRecord*   GetSpecRecord( void )               { return fStartSpecRec + fSpecCount; }
    scSpecRecord*   GetSpecRecord( long offset );

#ifdef _RUBI_SUPPORT
    void            SetAnnotations( scRubiArray* annot ){ fAnnotations = annot; }
    scRubiArray*    GetAnnotations( void )              { return fAnnotations; }
#endif

    void            SetFlowDir( const scFlowDir& fd )   { fInitialLine.SetFlowDir( fd ); }

//private:
                    // the previous para's data
    scContUnit*     fPrevPara;
    TypeSpec        fPrevSpec;
    scMuPoint       fOrigin;
    scLEADRefData   fPrevLead;

    scTextline*     fPrevline;
    scTextline*     fTextline;

                // the current paragraphs data
    scContUnit*     fPara;

                // the flow direction we are currently working with
    scFlowDir       fFlowDir;

                // this is set only at the begginning of the paragraph
    eTSCompLang     fBreakLang;

                // this is state that is maintained regarding widow/orphan and
                // lines hyphenated
    scParaColBreak  fBreakParams;
    eBreakType      fBreakType;

                // the number of columns this paragraph has snaked thru
    short           fColumnCount;
                // lines before the column break
    short           fLinesBefore;
                // lines after column break
    short           fLinesAfter;
    MicroPoint      fComputedLen;

                    // rag setting of the current line, used for positioning
                    // in flex columns
    eTSJust         fRagSetting;

                    // the init lead state of the line
    scLEADRefData   fInitialLead;

                    // the lead state after line breaking
    scLEADRefData   fEndLead;

                    // the letterspace of the line - set in the line breaker
    GlyphSize       fLetterSpace;

                    // the start and end angle of the glyphs ( pseudo-obliquing )
    scAngle         fStartAngle;
    scAngle         fEndAngle;

    eColShapeType   fColShapeType;
    MicroPoint      fBaselineJump;

    scFlowDir       fFlowDir;

private:
                    // the spec with the max lead on the line
                    // !!!!!NOTE: this should only be set in PARAInit
                    // or in the line breaker
    TypeSpec        initialSpec_;
    TypeSpec        fMaxLeadSpec;
};

/* ==================================================================== */

class scPARARefData {
public:
                    scPARARefData();

    void            PARAInit( scContUnit*,
                              const scFlowDir& );
    void            PARAFini( void );

    void            SaveData( void );
    void            RestoreData( void );

    Bool            ComposeLine( DCState& );

    Bool            AdjustLead( void ) const;

    void            SetColumn( scColumn* col )
                    {
                        column_ = col;
                    }
    scColumn*       GetColumn() const
                    {
                        return column_;
                    }

                    // the line data before we call the line breaker
    scLINERefData   fInitialLine;

                    // the information after we call the line breaker
    scLINERefData   fComposedLine;


        // this resets the InitialLineData between calls to the linebreaker
    void            SetLineData( Bool );


    void            SetPrevPara( scContUnit* p )           { fPrevPara = p; }
    scContUnit*     GetPrevPara( void )                { return fPrevPara; }
    MicroPoint       GetComputedLen( void )              { return fComputedLen; }

    void             SetRagSetting( eTSJust rag )        { fRagSetting = rag; }
    eTSJust          GetRagSetting( void )               { return fRagSetting; }

    void             SetInitialLead( const scLEADRefData& lrd )  { fInitialLead = lrd; }
    scLEADRefData&   GetInitialLead( void )                      { return fInitialLead; }

    void             SetEndLead( const scLEADRefData p ) { fEndLead = p; }
    scLEADRefData&   GetEndLead( void )                  { return fEndLead; }

    void             SetStartAngle( scAngle p )          { fStartAngle = p; }
    scAngle          GetStartAngle( void )               { return fStartAngle; }

    void             SetEndAngle( scAngle p )            { fEndAngle = p; }
    scAngle          GetEndAngle( void )                 { return fEndAngle; }

    void             SetColShapeType( eColShapeType p )  { fColShapeType = p; }
    eColShapeType    GetColShapeType( void )             { return fColShapeType; }

    void             SetBaselineJump( MicroPoint p )     { fBaselineJump = p; }
    MicroPoint       GetBaselineJump( void )             { return fBaselineJump; }


                     // origin of the line, this is orginally set in
                     // COLGetStrip and may be further modified in
                     // the LineBreaker
    scMuPoint        fOrg;
                     // this is used in COLGetStrip
    MicroPoint       fBaseline;

                     //
                     // the backing store set in the para initializer
                     //
    CharRecordP      fCharRecs;
                     // set in the para reformatter
    long             fStartCharOffset;
                     // set in the linebreaker
    long             fCharCount;


    scSpecRecord*    fSpecRec;
                     // set in the para reformatter
    long             fStartSpecRunOffset;
                     // set in the linebreaker
    long             fSpecRunCount;

#ifdef _RUBI_SUPPORT
                     // the character annotations ( rubi or hyper-text )
    scRubiArray*     fAnnotations;
#endif


    // somewhere along in here we would add the unpositioned glyphs
    // and the positioned glyphs hooks


                     // set in the LineBreaker
    scXRect          fInkExtents;            // extents on ink

                     // set before calling COLGetStrip and may be
                     // modified in the LineBreaker ( with a spec change )
    scXRect          fLogicalExtents;        // extents of em square

                     // length of last line, used for aesthetic rag
    MicroPoint       fLastLineLen;

                     // measure of the line as reported by COLGetStrip,
                     // may be modified in the LineBreaker
    MicroPoint       fMeasure;

                     // actual length after line breaking
```

```
                        fMaxLeadSpec( 0 ),
#ifdef _RUBI_SUPPORT
                        fAnnotations( 0 ),
#endif
                        fLastLineLen( 0 ),
                        fMeasure( 0 ),
                        fComputedLen( 0 ),
                        fRagSetting( eNoRag ),
                        fStartAngle( 0 ),
                        fEndAngle( 0 ),
                        fColShapeType( eNoShape ),
                        fBaselineJump( 0 ){}

                    // zero out all the line values
    void            Init( const scFlowDir& );
    void            xxInit( void );

    Bool            IsHorizontal( void ) const          { return fFlowDir.IsHorizontal(); }
    Bool            IsVertical( void ) const            { return fFlowDir.IsVertical(); }

    void            SetFlowDir( const scFlowDir& fd )   { fFlowDir = fd; }
    const scFlowDir& GetFlowDir( void ) const           { return fFlowDir; }

    void            SetOrg( const scMuPoint& p )        { fOrg = p; }
    scMuPoint&      GetOrg( void )                      { return fOrg; }

    void            SetBaseline( MicroPoint bl )        { fBaseline = bl; }
    MicroPoint      GetBaseline( void ) const           { return fBaseline; }

    void            SetCharacters( CharRecordP chrec )  { fCharRecs = chrec; }
    CharRecordP     GetCharacters( void ) const         { return fCharRecs; }

    void            SetStartCharOffset( long n )        { fStartCharOffset = n; }
    long            GetStartCharOffset( void ) const    { return fStartCharOffset; }

    void            SetCharCount( long n )              { fCharCount = n; }
    long            GetCharCount( void ) const          { return fCharCount; }

    long            GetEndCharOffset( void ) const      { return fStartCharOffset + fCharCount; }

    void            SetSpecRec( scSpecRecord* sr )      { fSpecRec = sr; }
    scSpecRecord*   GetSpecRec( void )                  { return fSpecRec; }

    void            SetStartSpecRunOffset( long n )     { fStartSpecRunOffset = n; }
    long            GetStartSpecRunOffset( void )       { return fStartSpecRunOffset; }

    void            SetSpecRunCount( long n )           { fSpecRunCount = n; }
    long            GetSpecRunCount( void )             { return fSpecRunCount; }

    void            SetMaxLeadSpec( TypeSpec p )        { fMaxLeadSpec = p; }
    TypeSpec        GetMaxLeadSpec( void )              { return fMaxLeadSpec; }

    void            SetInitialSpec( TypeSpec& p )       { initialSpec_ = p; }
    TypeSpec        GetInitialSpec( void )              { return initialSpec_; }

#ifdef _RUBI_SUPPORT
    void            SetAnnotations( scRubiArray* ra )   { fAnnotations = ra; }
    scRubiArray*    GetAnnotations( void )              { return fAnnotations; }
#endif

    void            SetInkExtents( const scXRect& r )   { fInkExtents = r; }
    scXRect&        GetInkExtents( void )               { return fInkExtents; }

    void            SetLogicalExtents( const scXRect& r )   { fLogicalExtents = r; }
    scXRect&        GetLogicalExtents( void )               { return fLogicalExtents; }

    void            SetLastLineLen( MicroPoint len )    { fLastLineLen = len; }
    MicroPoint      GetLastLineLen( void )              { return fLastLineLen; }

    void            SetMeasure( MicroPoint m )          { fMeasure = m; }
    MicroPoint      GetMeasure( void )                  { return fMeasure; }

    void            SetComputedLen( MicroPoint len )    { fComputedLen = len; }
    MicroPoint          fAboveLead;
    MicroPoint          fBelowLead;

    MicroPoint          fExternalLead;
};


/* ==================================================================== */


class DCState {
public:
    Bool                fDCSet;
    DropCapInfo         fDCInfo;
    scColumn*           fColumn;
    MicroPoint          fDCLastBaseline;

            DCState() :
                fDCSet( false ),
                fColumn( 0 ),
                fDCLastBaseline( 0 ){}

    void    SetColumn( scColumn *col ) { fColumn = col; }
    scColumn*   GetColumn( void ) const { return fColumn; }
};

/* ==================================================================== */
// redisplay information

class scRedisplayStoredLine {
public:
                    scRedisplayStoredLine( int lines );
                    ~scRedisplayStoredLine();

                    scRedisplayStoredLine() :
                        fStoredData( 0 ),
                        fStoredLines( 0 ),
                        fUsingStoredData( false ),
                        fData( 0 ),
                        fNumItems( 0 ){}


    void            LineListInit( int lines = 200 );
    void            LineListFini( void );

    void            SaveLineList( scColumn* );
    void            LineListChanges( scColumn*, const scXRect&, scRedispList* );

private:
    scTextline*     fStoredData;        /* the buffer */
    ushort          fStoredLines;       /* # of lines the buffer can store */
    Bool            fUsingStoredData;   /* are we using the buffer or
                                         * a larger temp buf
                                         */
    scXRect         fOrgExtents;
    scTextline*     fData;
    ushort          fNumItems;
};

/* ==================================================================== */

class scLINERefData {
public:
                    scLINERefData() :
                        fOrg( 0, 0 ),
                        fBaseline( 0 ),
                        fCharRecs( 0 ),
                        fStartCharOffset( 0 ),
                        fCharCount( 0 ),
                        fSpecRec( 0 ),
                        fStartSpecRunOffset( 0 ),
                        fSpecRunCount( 0 ),
                        initialSpec_( 0 ),
                dcMaxY( 0 ),
                dcLineOrgChange( 0 ),
                dcHChange( 0 ),
                dcVMax( 0 ){}
} ;


Bool            DCCompute( DropCapInfo&,
                           TypeSpec&,          // para spec
                           TypeSpec&,          // character spec
                           MicroPoint,
                           MicroPoint,
                           UCS2 );


/* ================================================================= */

class scLEADRefData {
public:
                        scLEADRefData() :
                                fFlow( eRomanFlow ),
                                fAboveLead( 0 ),
                                fBelowLead( 0 ),
                                fExternalLead( 0 ){}

                        scLEADRefData( const scLEADRefData& );
                        scLEADRefData( const scFlowDir& fd )                 { Init( fd ); }
                        scLEADRefData( MicroPoint lead, const scFlowDir& fd )   { ComputeAboveBelow(
lead, fd ); }

    void                Init( const scFlowDir& fd )
                            {
                                fFlow = fd;
                                fAboveLead = 0;
                                fBelowLead = 0;
                                fExternalLead = 0;
                            }

    void                Set( MicroPoint lead );
    void                Set( MicroPoint lead, const scFlowDir& );
    void                Set( MicroPoint aboveLead, MicroPoint belowLead )       { SetAboveLead( abov
eLead ); SetBelowLead( belowLead ); }
    void                Set( MicroPoint aboveLead, MicroPoint belowLead, const scFlowDir& );

    MicroPoint          Compute( MicroPoint ptsize, MicroPoint lead, eFntBaseline baseline );
    void                ComputeAboveBelow( MicroPoint lead, const scFlowDir& );

    void                SetFlow( const scFlowDir& fd )  { fFlow = fd; }
    const scFlowDir&    GetFlow( void ) const       { return fFlow; }

    MicroPoint          GetLead( void ) const
                            {
                                return fAboveLead + fBelowLead + fExternalLead;
                            }

    void                SetAboveLead( MicroPoint lead ) { fAboveLead = lead; }
    MicroPoint          GetAboveLead( void ) const      { return fAboveLead; }

    void                SetBelowLead( MicroPoint lead ) { fBelowLead = lead; }
    MicroPoint          GetBelowLead( void ) const      { return fBelowLead; }

    void                SetExternalSpace( MicroPoint space )
                            {
                                fExternalLead = space;
                            }
    MicroPoint          GetExternalSpace( ) const
                            {
                                return fExternalLead;
                            }
private:
    scFlowDir           fFlow;
                            // the above and below are a function of the
                            // ptsize and the baseline and external lead
```

```
/****************************************************************************

    File:       screfdat.h


    $Header: /Projects/Toolbox/ct/SCREFDAT.H 3     5/30/97 8:45a Wmanis $

    Contains:   Contans structs and classes for performing
                reformattng of text.

    Written by: Manis

    Copyright (c) 1989-1994 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.


****************************************************************************/


#ifndef _H_SCREFDAT
#define _H_SCREFDAT

#include "sctbobj.h"
#include "scspcrec.h"

class scRubiArray;
class PrevParaData;
class scContUnit;
class scColumn;
class scTextline;
class HRgn;
class scRedispList;

typedef enum eBreakTypes {
    eUndefinedBreak = -1,
    eNoBreak,
    eEndStreamBreak,
    eColumnBreak,
    eHyphBreak,
    eCharBreak,
    eQuadBreak,
    eSpaceBreak,
    eHardHyphBreak,
    eParaBreak,                 /* start of a paragraph */
    eStartColBreak
} eBreakType;

/* ================================================================== */

struct DropCapInfo {
    DCPosition  dcPosition;         // dc positioning info from spec
    MicroPoint  dcMinX,             // left char extent from the dc origin
                dcMinY,             // top char extent from the dc origin
                dcMaxX,             // right char extent from the dc origin
                dcMaxY,             // bottom char extent from dc origin
                dcLineOrgChange,    // the recomputed line origin
                dcHChange,          // horz offset from recomputed orgin
                dcVMax;             // until a baseline exceeds this values
                                    // readjust the line origins

            DropCapInfo() :
                dcMinX( 0 ),
                dcMinY( 0 ),
                dcMaxX( 0 ),
}

/**********************************************************************/
```

```
                    i++;

                if ( &s[i] >= endSliv )
                    goto bad;
                if ( s[i].fSLTop > testY ) {      /* y-level won't work */
                    if ( lead == 0 )
                        goto bad;
                    topY += lead;
                    leftX = fMaxBounds.x1 - fOrg.x;
                    s++;
                    break;
                }
                if ( s[i].fSLx1 > leftX ) {
                    leftX = s[i].fSLx1;
                    break;
                }

                if ( !RectInSliver( theRect, &s[i], leftX ) ) {

                    i++;
                    while ( &s[i] < endSliv && s[i].fSLTop <= testY
                                && !RectInSliver( theRect, &s[i], leftX ) )
                        i++;
                    if ( &s[i] >= endSliv )
                        goto bad;
                    if ( s[i].fSLTop > testY ) {      /* y-level won't work */
                        if ( lead == 0 )
                            goto bad;
                        topY += lead;
                        leftX = fMaxBounds.x1 - fOrg.x;
                        s++;
                        break;
                    }

                    scAssert( s[i].fSLTop <= testY &&
                            testY < s[i].fSLTop + vertInt &&
                            RectInSliver( theRect, &s[i], leftX ) );

                    leftX = s[i].fSLx1;             /* try this at same y-level */
                    break;

                } else {

                    if ( s[i].fSLx2 < bestX2 )
                        bestX2 = s[i].fSLx2;
                    if ( bottomY <= s[i].fSLTop + vertInt )
                        goto good;

                }

            }   /* inner for */

        }   /* the big else of the outer for loop */

    }   /* outer for loop */


good:
    {
        MicroPoint depth = theRect.Depth();
        theRect.x1 = leftX + fOrg.x;
        theRect.x2 = bestX2;


        theRect.y1 = topY + fOrg.y;
        theRect.y2 = theRect.y1 + depth;
    }
    return;

bad:
    theRect.x1 = theRect.x2 = 0;
    theRect.y1 = theRect.y2 = 0;
    return;
    leftX   -= fOrg.x;

    scAutoUnlock    h1( fSlivers );
    s = (Sliver *)*h1;

    endSliv = &s[fNumSlivers];
    vertInt = fVertInterval;

    for ( ; s < endSliv; ) {

        if ( s->fSLTop + vertInt <= topY ) {
            s++;
        }

        else if ( s->fSLTop > topY ) {
            if ( lead == 0 )
                goto bad;
            topY += lead;
            leftX = fMaxBounds.x1 - fOrg.x;

        } else if ( !RectInSliver( theRect, s, leftX ) ) {
            s++;
        } else {

            if ( s->fSLx1 > leftX )
                leftX = s->fSLx1;

            bestX2 = s->fSLx2;

            scAssert( s->fSLTop <= topY &&
                    topY < s->fSLTop + vertInt &&
                    s->fSLx1 <= leftX &&
                    leftX + theRect.Width() <= s->fSLx2 );

            /* Start position is reasonable; does it fit?   */
            /* Loop through sliver levels, to make sure     */
            /* the whole rectangle fits.                    */
            /* If we encounter a y-level with no fSlivers,   */
            /* increment sliver pointer past this level     */
            /* and try again (topY will catch via the main  */
            /* for loop (2nd clause) ).                     */
            /* If we fail at some y-level because of x-     */
            /* coordinates, look for the next possible      */
            /* leftX at that sliver level and try that      */
            /* (keeping the same topY).                     */
            /* If there is no next possible leftX, we have  */
            /* failed at this topY;                         */
            /* try the next one, BUT don't go too far       */
            /* ahead, since we may go down and to the left  */
            /* to find success -- so just increment topY by */
            /* the leading value.                           */

            testY = topY + vertInt;
            bottomY = topY + theRect.Depth();
            for ( i = 1; ; testY += vertInt ) {

                if ( &s[i] >= endSliv )
                    goto bad;
                if ( !SlivLE( vertInt, testY, bottomY ) )
                    goto good;

                while ( &s[i] < endSliv && s[i].fSLTop + vertInt <= testY )
                    i++;
                if ( &s[i] >= endSliv )
                    goto bad;
                if ( s[i].fSLTop > testY  ) {       /* y-level missing  */
                    s += i;
                    break;
                }

                while ( &s[i] < endSliv
                            && s[i].fSLTop <= testY
                                && s[i].fSLx2 <= leftX )
    scAutoUnlock    h2( r->fSlivers );
    Sliver*     s   = (Sliver*)*h2;

    int         i;

    for ( i = 0; i < r->fNumSlivers; i++ )
        InvertSliver( s[i], r->fVertInterval, appdc, drawrect, size, vertflow );
}

#endif

/* =================================================================== */

/* 1 if width of rectangle fits in sliver starting at max(x1,s->x1) */
/* 0 o.w.                                                           */

inline Bool RectInSliver( const scXRect&    theRect,
                          Sliver*           s,
                          MicroPoint        x1 )
{
    if ( s->fSLx1 > x1 )
        x1 = s->fSLx1;
    return( x1 + theRect.Width() <= s->fSLx2 );
}

/*******************************************************************/
/* 1 if sliverY is at the same (or a previous) sliver level as     */
/* floatingY; 0 o.w. SliverY is on a sliver boundary. FloatingY    */
/* is not.                                                         */

inline Bool SlivLE( MicroPoint vertInt,
                    MicroPoint sliverY,
                    MicroPoint floatingY )
{
    return( sliverY <= floatingY || floatingY <= sliverY + vertInt );
}

/*******************************************************************/
/* Given a rectangle (only size is given as input, not location),  */
/* a region, the top right corner of the last rectangle placed,    */
/* and a leading value, return the rectangle which has been placed */
/* in the first available location. Minimize y-position; within any */
/* given y-position, minimize x-position. If current y is          */
/* impossible, increment y by the leading value. If no position is */
/* available, return a rectangle with 0 width and depth.           */
/* The rectangle passed as input is altered, and is returned as    */
/* output. theRect.w on input specifies minimum width              */
/* SectRectRgn, upon finding a location, returns the widest        */
/* rectangle that will fit.                                        */


void HRgn::SectRect( scXRect&   theRect,
                     MicroPoint topY,
                     MicroPoint leftX,
                     MicroPoint lead )
{
    Sliver      *s;
    Sliver      *endSliv;
    MicroPoint  testY;
    MicroPoint  vertInt,
                bottomY,
                bestX2;
    int     i;

    if ( lead != 0 ) {
        for ( ; topY < fMaxBounds.y1; topY += ABS( lead )  )
            leftX = fMaxBounds.x1;
    }
    else
        topY = fMaxBounds.y1;

    topY    -= fOrg.y;
```

```
        scAssert ((size_t)(pbuf-rgnBuf) == sizeof(rgnBuf));

        byteSize    = r->fMaxSlivers * sizeof( Sliver );

        r->fSlivers = MEMAllocHnd( r->fMaxSlivers * sizeof( Sliver ) );

        scAutoUnlock    h2( r->fSlivers );
        s   = (Sliver*)*h2;

        int         i;
        int         n;
        uchar       sliverBuf[SCCB_SLIVER];

        n = r->fNumSlivers;

        for ( i = 0;  i < n; i++, s++ ) {

            ReadBytes( sliverBuf, ctxPtr, readFunc, sizeof( sliverBuf ) );
            pbuf = sliverBuf;
            pbuf = BufGet_long( pbuf, val, kIntelOrder );
            s->fSLx1    = val;
            pbuf = BufGet_long( pbuf, val, kIntelOrder );
            s->fSLx2    = val;
            pbuf = BufGet_long( pbuf, val, kIntelOrder );
            s->fSLTop   = val;

        }
    } catch ( ... ) {
        if ( rgnH ) {
            r = (HRgn*)scMemDeref( rgnH );
            MEMFreeHnd( r->fSlivers );
            MEMFreeHnd( rgnH );
        }
        throw;
    }

    return rgnH;

/* ==================================================================== */

#if SCDEBUG > 1

static void InvertSliver( const Sliver& sliver,
                          MicroPoint    interval,
                          APPDrwCtx     appdc,
                          HiliteFuncPtr drawrect,
                          const scSize& size,
                          int vertflow )
{
    scXRect rect;

    if ( vertflow ) {
        rect.Set( size.Width() - sliver.fSLTop,
                  sliver.fSLx1,
                  size.Width() - sliver.fSLTop + interval,
                  sliver.fSLx2 );
    }
    else
        rect.Set( sliver.fSLx1, sliver.fSLTop, sliver.fSLx2, sliver.fSLTop + interval );
    (*drawrect)( rect, appdc, false );
}

/* ==================================================================== */

void RGNInvertSlivers( const HRgnHandle rgnH,
                       APPDrwCtx        appdc,
                       HiliteFuncPtr    drawrect,
                       const scSize& size,
                       int vertflow )
{
    scAutoUnlock    h( rgnH );
    HRgn*       r   = (HRgn*)*h;
    n = (int)r->fNumSlivers;

    for ( i = 0;  i < n; i++, sliver++ ) {
        pbuf = sliverBuf;
        pbuf = BufSet_long( pbuf, sliver->fSLx1, kIntelOrder );
        pbuf = BufSet_long( pbuf, sliver->fSLx2, kIntelOrder );
        pbuf = BufSet_long( pbuf, sliver->fSLTop, kIntelOrder );

        WriteBytes( sliverBuf, ctxPtr, writeFunc, sizeof( sliverBuf ) );
    }

    return true;
}

/* ==================================================================== */

HRgnHandle RGNfromFile( APPCtxPtr       ctxPtr,
                        IOFuncPtr       readFunc,
                        int )
{
    HRgnHandle  rgnH;
    HRgn*       r = 0;
    Sliver*     s = 0;
    size_t      byteSize;
    Bool        tf = false;

    byteSize    = sizeof( HRgn );

    try {
        rgnH    = (HRgnHandle)MEMAllocHnd( byteSize );

        scAutoUnlock h( rgnH );

        r   = (HRgn*)*h;

        uchar           rgnBuf[SCCB_HRGN];
        const uchar*    pbuf = rgnBuf;

        ReadBytes( rgnBuf, ctxPtr, readFunc, sizeof( rgnBuf ) );

        ulong val;
        pbuf = BufGet_long( pbuf, val, kIntelOrder );
        r->fOrigBounds.x1   = val;
        pbuf = BufGet_long( pbuf, val, kIntelOrder );
        r->fOrigBounds.y1   = val;
        pbuf = BufGet_long( pbuf, val, kIntelOrder );
        r->fOrigBounds.x2   = val;
        pbuf = BufGet_long( pbuf, val, kIntelOrder );
        r->fOrigBounds.y2   = val;

        pbuf = BufGet_long( pbuf, val, kIntelOrder );
        r->fMaxBounds.x1    = val;
        pbuf = BufGet_long( pbuf, val, kIntelOrder );
        r->fMaxBounds.y1    = val;
        pbuf = BufGet_long( pbuf, val, kIntelOrder );
        r->fMaxBounds.x2    = val;
        pbuf = BufGet_long( pbuf, val, kIntelOrder );
        r->fMaxBounds.y2    = val;

        pbuf = BufGet_long( pbuf, val, kIntelOrder );
        r->fOrg.x   = val;
        pbuf = BufGet_long( pbuf, val, kIntelOrder );
        r->fOrg.y   = val;
        pbuf = BufGet_long( pbuf, val, kIntelOrder );
        r->fVertInterval    = val;

        ushort sval;
        pbuf = BufGet_short( pbuf, sval, kIntelOrder );
        r->fNumSlivers      = sval;
        pbuf = BufGet_short( pbuf, sval, kIntelOrder );
        r->fMaxSlivers      = sval;
    return HRgnSize( size );
}

/**************************************************************************/

void RGNGetExtents( const HRgnHandle rgnH, scXRect& xRect )
{
    const HRgn* r = (const HRgn*)scMemDeref( rgnH );

    xRect   = r->fOrigBounds;
}

/**************************************************************************/

MicroPoint RGNMaxDepth( const HRgnHandle rgnH )
{
    MicroPoint  depth   = LONG_MIN;
    const HRgn* r       = (HRgn*)scMemDeref( rgnH );

    depth   = r->fOrigBounds.y2;

    return depth;
}

/**************************************************************************/

#define SCCB_HRGN       48
#define SCCB_SLIVER     12

Bool RGNtoFile( APPCtxPtr       ctxPtr,
                IOFuncPtr       writeFunc,
                HRgnHandle      rgnH,
                int             size )
{
    HRgn*   r;
    Bool    tf = false;

    scAutoUnlock    h( rgnH );
    r   = (HRgn*)*h;

    scAssert( size == r->fNumSlivers );

    uchar           rgnBuf[SCCB_HRGN];
    uchar*          pbuf = rgnBuf;

    pbuf = BufSet_long( pbuf, r->fOrigBounds.x1, kIntelOrder );
    pbuf = BufSet_long( pbuf, r->fOrigBounds.y1, kIntelOrder );
    pbuf = BufSet_long( pbuf, r->fOrigBounds.x2, kIntelOrder );
    pbuf = BufSet_long( pbuf, r->fOrigBounds.y2, kIntelOrder );

    pbuf = BufSet_long( pbuf, r->fMaxBounds.x1, kIntelOrder );
    pbuf = BufSet_long( pbuf, r->fMaxBounds.y1, kIntelOrder );
    pbuf = BufSet_long( pbuf, r->fMaxBounds.x2, kIntelOrder );
    pbuf = BufSet_long( pbuf, r->fMaxBounds.y2, kIntelOrder );

    pbuf = BufSet_long( pbuf, r->fOrg.x, kIntelOrder );
    pbuf = BufSet_long( pbuf, r->fOrg.y, kIntelOrder );
    pbuf = BufSet_long( pbuf, r->fVertInterval, kIntelOrder );

    pbuf = BufSet_short( pbuf, (ushort)r->fNumSlivers, kIntelOrder );
    pbuf = BufSet_short( pbuf, (ushort)r->fMaxSlivers, kIntelOrder );

    scAssert ((size_t)(pbuf-rgnBuf) == sizeof(rgnBuf));

    WriteBytes( rgnBuf, ctxPtr, writeFunc, sizeof( rgnBuf ) );


    scAutoUnlock    h2( r->fSlivers );
    Sliver*         sliver = (Sliver*)*h2;

    int         i;
    int         n;
    uchar       sliverBuf[SCCB_SLIVER];
```

```
        scAutoUnlock    h3( rgnH );
        r            = (HRgn*)*h3;

        r->fOrg.x    = r->fOrg.y = 0;

        vertInt      = r->fVertInterval;
        numActives   = 0;
        numUsed      = 0;

        currentY     = Massage( edges->y1 , vertInt, -1 );

        for( ; currentY <= maxY; currentY += vertInt ) {
            numActives = RemInactive( actives, currentY, numActives );
            CalcNewX( actives, currentY, numActives );
            SortActives( actives, numActives );
            AddActive( edges, actives, currentY, numEdges, &numActives, &numUsed );
            r->ScanEdges( actives, currentY, numActives );
        }
    } catch ( ... ) {
        MEMFreeHnd( edgeH );
        MEMFreeHnd( activeH );
        MEMFreePtr( newVList );
        throw;
    }

    MEMFreeHnd( edgeH );
    MEMFreeHnd( activeH );
    MEMFreePtr( newVList );
}

/*************************************************************************/
/* HRgnFirstLinePos returns the y position of the first line within      */
/* the region which is an itegral value of leading from firstLinePos.    */

MicroPoint  HRgn::FirstLinePos( MicroPoint  firstLinePos,
                                MicroPoint  leading )
{
#undef FloatRgnFirstLine
#ifdef FloatRgnFirstLine
    return fMaxBounds.y1 - fOrg.y + firstLinePos;
#else
    if ( fMaxBounds.y1 - fOrg.y == 0 )
        return firstLinePos;

    if ( leading != 0 ) {
        while ( firstLinePos > fMaxBounds.y1 - fOrg.y )
            firstLinePos -= ABS( leading );
    }
    else
        firstLinePos = fMaxBounds.y1 - fOrg.y;

    return firstLinePos;
#endif
}

/*************************************************************************/

MicroPoint  RGNSliverSize( const HRgnHandle rgnH )
{
    HRgn*       r;
    MicroPoint  sliverSize;

    r           = (HRgn*)scMemDeref( rgnH );
    sliverSize  = r->fVertInterval;

    return sliverSize;
}

/*************************************************************************/

long RGNExternalSize( const HRgnHandle, long size )
{
        if ( v->fPointType == eStartPoint )
            polyrect.Invalidate();

        polyrect.x1 = MIN( v->x, polyrect.x1 );
        polyrect.y1 = MIN( v->y, polyrect.y1 );
        polyrect.x2 = MAX( v->x, polyrect.x2 );
        polyrect.y2 = MAX( v->y, polyrect.y2 );

        if ( v->fPointType == eStopPoint && ( !polyrect.Width() || !polyrect.Depth() ) )
             return false;

    } while( (v++)->fPointType != eFinalPoint );

    return ( !polyrect.Width() || !polyrect.Depth() );
}

/* ================================================================== */
/* PolyHRgn converts a polygon, represented by a list of vertices,  */
/* into a region, using scan conversion techniques.                 */
/* Destroys the previous structure of the region.                   */

void    PolyHRgn( HRgnHandle        rgnH,
                  const scVertex*   vList )
{
    const scVertex* v;
    MicroPoint  currentY,
                vertInt,
                maxY,
                secondY;
    int         numEdges,
                numActives,
                numUsed;

    scVertex*                newVList    = 0;
    HEdgeHandle              edgeH       = 0;
    volatile HEdgeHandle     activeH     = 0;
    HEdge *volatile          actives = 0;
    HEdge *volatile          edges   = 0;
    HRgn  *volatile          r       = 0;

    try {
        raise_if( rgnH == 0, scERRinput );
        SetEmptyHRgn( rgnH );


        raise_if( emptyPoly( vList ), scERRinput );     // test for invalid poly

            // vectorize the beziers
        BEZVectorizePoly( newVList, vList );
        if ( newVList )
            vList = newVList;

        edgeH = (HEdgeHandle)MEMAllocHnd( edge_growSize * sizeof( HEdge ) );
        numEdges = 0;

        SetPolyBounds( vList, rgnH );

        for ( maxY = LONG_MIN, v = vList; v->fPointType != eFinalPoint; v++ )
            maxY = MAX( maxY, v->y );

        for ( ; vList->fPointType != eFinalPoint; vList++ ) {
            if ( vList->fPointType != eStopPoint ) {
                ConstructEdge( vList, &edgeH, &numEdges, &secondY );
            }
        }

        activeH = MEMResizeHnd( activeH, MEMGetSizeHnd( activeH ) + MEMGetSizeHnd( edgeH) );

        scAutoUnlock    h1( edgeH );
        edges       = (HEdge*)*h1;

        scAutoUnlock    h2( activeH );
        actives     = (HEdge*)*h2;
```

```
                (*numUsed)++;
                continue;
            }

            CalcNewX( e, currentY, 1 );


            for ( i = *numActives - 1; i >= 0 && e->x < actives[i].x; i-- )
                ;
            i++;

            if ( i < *numActives )
                SCmemmove( &actives[i+1], &actives[i], ( *numActives - i ) * sizeof( HEdge ) );

            (*numActives)++;
            (*numUsed)++;
            actives[i] = *e;
        }
}

/************************************************************************/
/* ScanEdges scans the edges at each scan line, adding fSlivers to    */
/* the region accordingly.                                              */

void HRgn::ScanEdges( HEdge*        a,
                      MicroPoint    curY,
                      int           numActives )
{
    HEdge*      end;
    MicroPoint  prevX,
                left    = 0,            /* to get compiler to shut up */
                right   = 0;            /* to get compiler to shut up */
    Bool        state,
                readySliver;

    readySliver = false;
    state       = false;

    end = &a[numActives];
    for( ; a < end; a++ ) {
        if ( state == false ) {
            prevX   = a->x;
            state   = true;
        }
        else {
            state   = false;
            if ( readySliver && right == prevX )
                right   = a->x;
            else {
                if ( readySliver )
                    AddSliver( left, right, curY );
                else
                    readySliver = true;
                left    = prevX;
                right   = a->x;
            }
        }
    }

    if ( readySliver )
        AddSliver( left, right, curY );

    raise_if( state, scERRlogical );
}

/************************************************************************/

int emptyPoly( const scVertex* vlist )
{
    scXRect polyrect;

    const scVertex* v = vlist;
    do {
            if ( i + 1 < numActives )
                SCmemmove( &actives[i], &actives[i+1], ( numActives - i - 1 ) * sizeof( HEdge ) );
            numActives--;
        }

    return numActives;
}

/*************************************************************************/
/* CalcNewX, given a current y position and a list of edges,       */
/* calculates their new x position.                                */

static void CalcNewX( HEdge*        e,
                      MicroPoint    yPos,
                      int           num )
{
    int     i;
    REAL    temp;

    for ( i = num-1; i >= 0; i-- ) {
        temp    = (REAL)( yPos - e[i].y1 );
        temp    *= (REAL)( e[i].dx );
        temp    /= (REAL)( e[i].dy );
        e[i].x  = (MicroPoint)temp + e[i].x1;
    }
}

/*************************************************************************/
/* SortActives sorts the edges in the active list by their x position   */
/* at the current y-position of the scan line. A bubble sort is used    */
/* because the list is almost always in sorted order or is very close.  */

static void SortActives( HEdge* e,
                         int    numActives )
{
    int     i,
            last;
    Bool    badOrder;
    HEdge   temp;

    badOrder    = true;
    last        = numActives - 1;
    for ( ; badOrder == true; last--) {
        badOrder = false;
        for ( i = 0; i < last; i++ )
            if ( e[i].x > e[i+1].x ) {
                badOrder    = true;
                temp        = e[i];
                e[i]        = e[i+1];
                e[i+1]      = temp;
            }
    }
}


/*************************************************************************/
/* AddActive, given a new scan line position, adds new active edges */
/* to the active edge list.                                          */

static void AddActive( HEdge*       e,
                       HEdge*       actives,
                       MicroPoint   currentY,
                       int      numEdges,
                       int*     numActives,
                       int*     numUsed )
{
    int     i;
    HEdge*  end;

    end = &e[numEdges];
    for ( e += *numUsed; e < end && currentY >= e->y1; e++ ) {

        if ( currentY > e->y2 ) {
```

```
    if ( (*numEdges+1) * sizeof(HEdge) > MEMGetSizeHnd( *edgeHP ) )
        *edgeHP = MEMResizeHnd( *edgeHP, MEMGetSizeHnd( *edgeHP ) + sizeof( HEdge ) * edge_growSize
);

    if ( vList->y < vList[1].y ) {
        y1  = vList->y;
        y2  = vList[1].y;
        x   = vList->x;
        dx  = vList[1].x - vList->x;
        dy  = vList[1].y - vList->y;
    }
    else {
        y1  = vList[1].y;
        y2  = vList->y;
        x   = vList[1].x;
        dx  = vList->x - vList[1].x;
        dy  = vList->y - vList[1].y;
    }

/*    Determine which endpoints, if any, are tricky.    */
/*    If so, change that endpoint to shorten the        */
/*    edge by one MicroPoint.                           */

    for ( i = 1;    vList[i].fPointType != eStopPoint       &&
                    vList[i].fPointType != eFinalPoint  &&
                    vList[i].y == vList[i+1].y;         )
        i++;

    if ( vList[i].fPointType == eFinalPoint || vList[i].fPointType == eStopPoint )
        nextY = *secondY;
    else
        nextY = vList[i+1].y;

    if ( vList->y < vList[1].y && vList[1].y < nextY )
        y2--;
    if ( vList->y > vList[1].y && vList[1].y > nextY )
        y1++;

/*    Insert the edge and set the fields. */

    scAutoUnlock    h1( *edgeHP );
    edges    = (HEdge *)*h1;

    for (   i = *numEdges - 1;
            i >= 0 && ( y1 < edges[i].y1 || y1 == edges[i].y1 && x < edges[i].x );
            i-- )
        ;
    i++;

    if ( i < *numEdges )
        SCmemmove( &edges[i+1], &edges[i], ( *numEdges - i ) * sizeof( HEdge ) );

    edges[i].y1     = y1;
    edges[i].y2     = y2;
    edges[i].x1     = x;
    edges[i].x      = x;
    edges[i].dx     = dx;
    edges[i].dy     = dy;

    (*numEdges)++;
}

/*************************************************************************/
/* RemInactive removes inactive edges from the active edge list */

static int RemInactive( HEdge   *actives,
                             MicroPoint     currentY,
                             int         numActives )
{
    int         i;

    for ( i = numActives - 1; i >= 0; i-- )
        if ( actives[i].y2 < currentY ) {
```

```
    HRgn*       r;
    MicroPoint  left    = LONG_MAX;
    MicroPoint  right   = LONG_MIN;
    MicroPoint  top     = LONG_MAX;
    MicroPoint  bottom  = LONG_MIN;

    for ( ; vList->fPointType != eFinalPoint; vList++ ) {
        left    = MIN( left, vList->x );
        right   = MAX( right, vList->x );
        top     = MIN( top, vList->y );
        bottom  = MAX( bottom, vList->y );
    }

    r       = (HRgn*)scMemDeref( rgnH );

    r->fOrigBounds.Set( left, top, right, bottom );
}

/*************************************************************************/
/* ConstructEdge inserts a new edge from the vertex list into the edge   */
/* list. The edge list is sorted in increasing order of the lowest       */
/* y point of the edge. Within equal minimum y values, edges are         */
/* sorted in increasing x order, using the x value of the vertex         */
/* with the minimum y value.                                             */
/* The edge that is inserted connects the vertices vList[0]              */
/* and vList[1].                                                         */
/* It is important to note that the ordering of vList depends on         */
/* how we trace the edges, so vList[0] is encountered before             */
/* vList[1]. This is not true, however, of e->fSLTop1 and e->fSLTop2. The           */
/* point ( e->fSLx1, e->fSLTop1 ) is defined as the endpoint with lowest        */
/* y-value, regardless of whether it was encountered first.              */
/* "tricky" endpoints are those which will cause problems for the        */
/* scanning algorithm if they are included as points along two          */
/* two edges (which they in fact are). "nextY" is the y-value            */
/* of vList[2], and is used to determine whether vList[1] is a           */
/* "tricky" endpoint. We can't simply use "vList[2]" for two             */
/* reasons: first, in the case of a horizontal edge that looks           */
/* like this:
              |_
                |,                                                       */
/* we must skip over one of the points which have equal y-value.         */
/* Second, when vList[1] is a stopPoint, we must use the point           */
/* right after the startPoint as our nextY. "secondY" is used to         */
/* keep track of this over a series of calls to ConstructEdge().         */
/* When an endpoint is found to be tricky, it is changed to shorten      */
/* the edge by one MicroPoint.                                           */

static void ConstructEdge( const scVertex*  vList,
                           HEdgeHandle*     edgeHP,
                           int*             numEdges,
                           MicroPoint*      secondY )
{
    HEdge*      edges;
    MicroPoint  y1,
                x,
                nextY;
    MicroPoint  y2,
                dx,
                dy;
    int         i;

    if ( vList->fPointType == eStartPoint ) {
        for (   i = 0; vList[i].y == vList[i+1].y   &&
                vList[i+1].fPointType != eStopPoint &&
                vList[i+1].fPointType != eFinalPoint; i++ )
              ;
        *secondY = vList[i+1].y;
    }

    if ( vList->y == vList[1].y )
        return;

//  if ( *numEdges * sizeof( HEdge ) == MEMGetSizeHnd( *edgeHP ) )
```

```
    DisposeHRgn( newRH );
}

/*************************************************************************/
/* YShrinkHRgn shrinks the region a distance of dy vertically.     */
/* dy is positive.                                                  */

static void YShrinkHRgn( HRgnHandle rgnH,
                         MicroPoint dy )
{
    volatile HRgnHandle         rH1;
    volatile HRgnHandle         rH2;
    volatile HRgnHandle         rH3;
    HRgn*                       r;
    MicroPoint                  offset,
                                totalOffset;
    MicroPoint                  sliverSize;

    scAutoUnlock    h1( rgnH );
    r           = (HRgn *)*h1;

    try {
        sliverSize  = r->fVertInterval;

        rH1 = NewHRgn( sliverSize );

        rH2 = NewHRgn( sliverSize );

        rH3 = NewHRgn( sliverSize );


        for ( offset = sliverSize, totalOffset = 0; ; ) {

            CopyHRgn( rH1, rgnH );

            CopyHRgn( rH2, rgnH );

            TranslateHRgn( rH1, 0L, offset );
            TranslateHRgn( rH2, 0L, - offset );

            SectHRgn( rH1, rgnH, rH3);
            SectHRgn( rH2, rH3, rgnH );

            totalOffset += offset;

            if ( totalOffset == dy )
                break;

            offset = 3 * offset;
            if ( totalOffset + offset > dy )
                offset = dy - totalOffset;

            scAssert( totalOffset + offset <= dy );
        }
    }
    catch ( ... ) {
        DisposeHRgn( rH1 );
        DisposeHRgn( rH2 );
        DisposeHRgn( rH3 );
        throw;
    }

    DisposeHRgn( rH1 );
    DisposeHRgn( rH2 );
    DisposeHRgn( rH3 );
}

/************************************************************************/
/* SetPolyBounds sets the bounding rectangle of the polygon                  */

static void SetPolyBounds( const scVertex*  vList,
                           HRgnHandle       rgnH )
{
    ~scAutoUnlock2( )
        {
            MEMUnlockHnd( handle_ );
        }

    void    *operator *()
                {
                    return scMemDeref( handle_ );
                }

private:
    scMemHandle& handle_;
};


static void YGrowHRgn( HRgnHandle   rgnH,
                       MicroPoint   dy )
{
    HRgnHandle  newRH;
    HRgn*       r;
    HRgn*       newR;
    Sliver*     s;
    Sliver*     newS;
    MicroPoint  yPos,
                j;
    int         i;

    try {
        scAutoUnlock    h1( rgnH );
        r = (HRgn*)*h1;

        newRH = NewHRgn( r->fVertInterval );

        scAutoUnlock    h3( newRH );
        newR    = (HRgn*)*h3;

        {
            scAutoUnlock2   h2( r->fSlivers );
            s       = (Sliver*)*h2;

            scAutoUnlock2   h4( newR->fSlivers );
            newS    = (Sliver*)*h4;

            newR->fOrg.x        = r->fOrg.x;
            newR->fOrg.y        = r->fOrg.y;
            newR->fVertInterval = r->fVertInterval;

            for ( i = 0; i < r->fNumSlivers; i++ ) {
                yPos = s[i].fSLTop;

                if ( r->IsBorder( i, -1 ) ) {
                    for ( j = dy; j < 0; j += r->fVertInterval )
                        newR->CorpSliver( &newS, s[i].fSLx1, s[i].fSLx2, yPos + j );
                }

                newR->CorpSliver( &newS, s[i].fSLx1, s[i].fSLx2, yPos );

                if ( r->IsBorder( i, 1 ) ) {
                    for ( j = r->fVertInterval; j <= -dy; j += r->fVertInterval )
                        newR->CorpSliver( &newS, s[i].fSLx1, s[i].fSLx2, yPos + j );
                }
            }

            newR->SetBounds( );
        }
        CopyHRgn( rgnH, newRH );
        SCDebugTrace( 0, "done" );
    }
    catch( ... ) {
        DisposeHRgn( newRH );
        throw;
    }
```

```
            right = s[i].fSLx2;

    i--;
    InsertSliver( left, right, yPos, sPP, start, i );
}

/****************************************************************************/
/* Like GetBorders, but boolean; returns 1 if GetBorders would return       */
/* nonzero value for *num (the number of borders).                          */

Bool HRgn::IsBorder( int    pos,
                     int    tb )
{
    Sliver*     s;
    int         i;
    MicroPoint  left,
                right,
                yPos;
    Bool        uncovered;

    scAutoUnlock    h1( fSlivers );
    s       = (Sliver *)*h1;

    i       = pos;
    left    = s[i].fSLx1;
    right   = s[i].fSLx2;
    yPos    = s[i].fSLTop + tb * fVertInterval;

    if ( tb == -1 ) {
        for ( ; i >= 0 && s[i].fSLTop > yPos; i-- )
            ;
        if ( i < 0 || s[i].fSLTop != yPos ) {
            return true;
        }
        for ( ; i >= 0 && s[i].fSLTop == yPos; i-- )
            ;
        i++;
    }
    else {
        for ( ; i < fNumSlivers && s[i].fSLTop < yPos; i++ )
            ;
        if ( i == fNumSlivers || s[i].fSLTop != yPos ) {
            return true;
        }
    }

    uncovered = true;
    for ( ; i < fNumSlivers && s[i].fSLTop == yPos && s[i].fSLx1 <= right; i++ ) {  /* process fSliv
ers at this yLevel  */
        if ( s[i].fSLx2 <= left )
            continue;
        if ( s[i].fSLx1 < right )
            uncovered = false;
        if ( s[i].fSLx1 > left || s[i].fSLx2 < right ) {
            return true;
        }
    }

    return uncovered;
}

/****************************************************************************/
/* YGrowHRgn expands the region a distance of dy vertically.        */
/* dy is negative.                                                  */

class scAutoUnlock2 {
public:
    scAutoUnlock2( scMemHandle& hnd ) :
        handle_( hnd )
        {
            MEMLockHnd( handle_ );
        }
```

```
/* amid potentially overlapping sliver neighbors.                    */

void HRgn::CorpSliver( Sliver**     sPP,
                       MicroPoint   left,
                       MicroPoint   right,
                       MicroPoint   yPos )
{
    Sliver* s;
    int     i;
    int     lo,
            hi;
    Bool    found;
    int     start;

    s = *sPP;

    if ( fNumSlivers == 0 ) {
        InsertSliver( left, right, yPos, sPP, 0, -1 );
        return;
    }

    lo      = 0;
    hi      = fNumSlivers - 1;
    found   = false;
    for ( i = (hi+lo)/2; lo <= hi; i = (hi+lo)/2 ) {
        if ( yPos < s[i].fSLTop )
            hi = i - 1;
        else if ( yPos > s[i].fSLTop )
            lo = i + 1;
        else {
            found = true;
            break;
        }
    }

    if ( !found ) {      /* don't remove, just insert   */
        if ( i < 0 )
            i = 0;
        else if ( i < fNumSlivers && s[i].fSLTop < yPos )
            i++;

        scAssert( i == fNumSlivers || s[i].fSLTop > yPos );

        InsertSliver( left, right, yPos, sPP, i, i-1 );
        return;
    }

    scAssert( i < fNumSlivers && yPos == s[i].fSLTop );

                    /* find the right place to start.   */
    if ( left <= s[i].fSLx2 ) {
        for ( ; i >= 0 && s[i].fSLTop == yPos && left <= s[i].fSLx2; i-- )
            ;
        i++;
    } else {
        for ( ; i < fNumSlivers && s[i].fSLTop == yPos && left > s[i].fSLx2; i++ )
            ;
    }

    if ( i == fNumSlivers || yPos < s[i].fSLTop ) {
        InsertSliver( left, right, yPos, sPP, i, i-1 );
        return;
    }

    scAssert( i < fNumSlivers && yPos == s[i].fSLTop && left <= s[i].fSLx2 );

    start = i;

    if ( left > s[i].fSLx1 )
        left = s[i].fSLx1;

    for ( ; i < fNumSlivers && yPos == s[i].fSLTop && right >= s[i].fSLx1; i++ )
        if ( right < s[i].fSLx2 )
        scAutoUnlock    h1( rgnH );
        r = (HRgn *)*h1;

        newRH = NewHRgn( r->fVertInterval );

        scAutoUnlock    h2( newRH );
        newR    = (HRgn *)*h2;

        {
            scAutoUnlock    h3( r->fSlivers );  // lock for the duration of AddSliver calls
            s       = (Sliver *)*h3;

            newR->fOrg.x        = r->fOrg.x;
            newR->fOrg.y        = r->fOrg.y;
            newR->fVertInterval = r->fVertInterval;

            end = &s[r->fNumSlivers];
            for ( ; s < end; s++ ) {
                if ( s->fSLx2 - s->fSLx1 > dx * 2 )
                    newR->AddSliver( s->fSLx1 + dx + r->fOrg.x, s->fSLx2 - dx + r->fOrg.x, s->fSLTop
 + r->fOrg.y );
            }
        }
        CopyHRgn( rgnH, newRH );
    }
    catch( ... ) {
        DisposeHRgn( newRH );
        throw;
    }

    DisposeHRgn( newRH );
}

/**********************************************************************/
/* InsertSliver inserts the sliver into the given region at the     */
/* position "start". The fSlivers in positions "start" to "end" are */
/* removed from the region. If end < start, no fSlivers are removed.    */

void HRgn::InsertSliver( MicroPoint left,
                         MicroPoint right,
                         MicroPoint y,
                         Sliver**   sPP,
                         int        start,
                         int        end )
{
    Sliver* s;

    s = *sPP;
    if ( end < start ) {            /* don't remove, just insert    */
        if ( fNumSlivers == fMaxSlivers ) {
            MEMUnlockHnd( fSlivers );
            fSlivers = MEMResizeHnd( fSlivers, MEMGetSizeHnd( fSlivers ) + sliver_growSize * sizeof(
 Sliver ) );
            s   = *sPP  = (Sliver *)MEMLockHnd( fSlivers );
            fMaxSlivers += sliver_growSize;
        }
        if ( start < fNumSlivers )
            SCmemmove( &s[start+1], &s[start], ( fNumSlivers - start ) * sizeof( Sliver ) );
        fNumSlivers++;
    }
    else if ( start < end ) {       /* remove more than one */
        if ( end + 1 < fNumSlivers )
            SCmemmove( &s[start+1], &s[end+1], ( fNumSlivers - end - 1 ) * sizeof( Sliver ) );
        fNumSlivers -= end - start;
    }

    (&s[start])->SetSliver( left, right, y );
}

/**********************************************************************/
/* CorpSliver, unlike its simplistic counterpart "AddSliver",       */
/* judiciously incorporates the sliver into the given region,       */
/* displaying consummate tact in smoothly integrating the sliver    */
```

```
    DisposeHRgn( vertRgnH );
}

/************************************************************************/
/* XGrowHRgn expands the region a distance of dx horizontally.        */
/* dx is negative.                                                     */

static void XGrowHRgn( HRgnHandle   rgnH,
                       MicroPoint   dx )
{
    volatile HRgnHandle newRH;
    HRgn*               r;
    HRgn*               newR;
    Sliver*             s;
    Sliver*             end;
    MicroPoint          left,
                        right,
                        yPos;

    try {
        scAutoUnlock    h1( rgnH );
        r = (HRgn*)*h1;

        newRH = NewHRgn( r->fVertInterval );

        scAutoUnlock    h2( newRH );
        newR    = (HRgn*)*h2;

        {
            scAutoUnlock    h3( r->fSlivers );
            s       = (Sliver*)*h3;

            newR->fOrg.x        = r->fOrg.x;
            newR->fOrg.y        = r->fOrg.y;
            newR->fVertInterval = r->fVertInterval;

            end = &s[r->fNumSlivers];
            for ( ; s < end; ) {
                yPos    = s->fSLTop;
                left    = s->fSLx1 + dx;
                right   = s->fSLx2 - dx;

                for ( s++; s < end && s->fSLTop == yPos && right >= s->fSLx1 + dx; s++ )
                    right = s->fSLx2 - dx;

                newR->AddSliver( left + r->fOrg.x, right + r->fOrg.x, yPos + r->fOrg.y );
            }
        }
        CopyHRgn( rgnH, newRH );
    }
    catch ( ... ) {
        DisposeHRgn( newRH );
        throw;
    }

    DisposeHRgn( newRH );
}

/************************************************************************/
/* XShrinkHRgn shrinks the region a distance of dx horizontally.        */
/* dx is positive.                                                       */

static void XShrinkHRgn( HRgnHandle rgnH,
                         MicroPoint dx )
{
    volatile HRgnHandle newRH;
    HRgn*               r;
    HRgn*               newR;
    Sliver*             s;
    Sliver*             end;

    try {
        vertInt = 2 * r->fVertInterval;
    }
    // we should test the validity of the inset region, if it has no size or negative
    // size we should raise an exception

    if ( ( dx < 0 && dy > 0 ) || ( dx > 0 && dy < 0 ) ) {
        OldInsetHRgn( rgnH, dx, dy );
    }
    else if ( dx != dy || bestOption == 0 ) {
        PlainInsetHRgn( rgnH, dx, dy );
    }
    else {
        if ( dx > 0 ) {
            for ( distance = vertInt; distance <= dx; distance += vertInt ) {
                PlainInsetHRgn( rgnH, vertInt, vertInt );
            }
            distance    -= vertInt;
            PlainInsetHRgn( rgnH, dx - distance, dx - distance );
        }
        else if ( dx < 0 ) {
            for ( distance = - vertInt; distance >= dx; distance -= vertInt ) {
                PlainInsetHRgn( rgnH, - vertInt, - vertInt );
            }
            distance    += vertInt;
            PlainInsetHRgn( rgnH, dx - distance, dx - distance );
        }
    }
}

/***************************************************************************/
/* IMPORTANT NOTE: This works only when sign( dx ) = sign( dy ) */

static void PlainInsetHRgn( HRgnHandle  rgnH,
                            MicroPoint  dx,
                            MicroPoint  dy )
{
    volatile HRgnHandle vertRgnH;
    HRgn*               r;
    MicroPoint          vertInt;

    if ( dx == 0 && dy == 0 )
        return;

    try {
        r       = (HRgn*)scMemDeref( rgnH );
        vertInt = r->fVertInterval;

        vertRgnH = NewHRgn( vertInt );

        CopyHRgn( vertRgnH, rgnH );

        if ( dx < 0 )
            XGrowHRgn( rgnH, dx );
        else if ( dx > 0 )
            XShrinkHRgn( rgnH, dx );

        r   = (HRgn*)scMemDeref( vertRgnH );
        dy  = PointMassage( dy, r->fVertInterval );

        if ( dy < 0)
            YGrowHRgn( vertRgnH, dy );
        else if ( dy > 0 )
            YShrinkHRgn( vertRgnH, dy );

        if ( dx > 0 || dy > 0 )
            SectHRgn( rgnH, vertRgnH, rgnH );
        else
            UnionHRgn( rgnH, vertRgnH, rgnH );
    }
    catch ( ... ) {
        DisposeHRgn( vertRgnH );
        throw;
    }
```

```
        if (    a->fSLx1 + aX != b->fSLx1 + bX  ||
                a->fSLx2 + aX != b->fSLx2 + bX  ||
                a->fSLTop + aY  != b->fSLTop + bY       )
        {
            same = false;
        }
    }

    return same;
}


/************************************************************************/
/* EmptyHRgn returns non-zero if the region contains at least   */
/* one sliver, 0 o.w.                                           */

Bool EmptyHRgn( HRgnHandle  rgnH )
{
    HRgn*   r;
    int     num;

    r   = (HRgn *)scMemDeref( rgnH );
    num = r->fNumSlivers;

    return num == 0;
}


/************************************************************************/
/* InsetHRgn shrinks or expands the region. All points on the region    */
/* boundary are moved inwards a distance of dy vertically and dx        */
/* horizontally; if dx or dy is negative, the points are moved outwards */

static void OldInsetHRgn ( HRgnHandle   rgnH,
                           MicroPoint   dx,
                           MicroPoint   dy )
{
    HRgn  *r;

    if ( dx < 0 )
        XGrowHRgn( rgnH, dx );
    else if ( dx > 0 )
        XShrinkHRgn( rgnH, dx );

    r   = (HRgn*)scMemDeref( rgnH );
    dy  = PointMassage( dy, r->fVertInterval );

    if ( dy < 0)
        YGrowHRgn( rgnH, dy );
    else if ( dy > 0 )
        YShrinkHRgn( rgnH, dy );
}

/************************************************************************/

void InsetHRgn( HRgnHandle  rgnH,
                MicroPoint  dx,
                MicroPoint  dy,
                Bool        bestOption )
{
    MicroPoint  vertInt,
                distance;
    HRgn*       r;

    {
        scAutoUnlock    h1( rgnH );
        r = (HRgn *)*h1;

        r->fOrigBounds.x1   = r->fOrigBounds.x1 + dx;
        r->fOrigBounds.y1   = r->fOrigBounds.y1 + dy;
        r->fOrigBounds.x2   = r->fOrigBounds.x2 - dx;
        r->fOrigBounds.y2   = r->fOrigBounds.y2 - dy;
```

```
}


/***************************************************************************/
/* RectInHRgn returns 1 if the rectangle intersects the region, 0 o.w.  */
/*  Note: RectInHRgn will sometimes return 1 when the rectangle merely  */
/*  intersects the region's enclosing rectangle. If you need to know    */
/*  whether a rectangle intersects the actual region, use RectHRgn to   */
/*  set the rectangle to a region, and call SectHRgn to see whether     */
/*  they intersect. If the result of SectHRgn is an empty region,       */
/*  they don't intersect.                                                */

Bool RectInHRgn( const HRgnHandle   rgnH,
                 const scXRect&     rec )
{
    HRgn*   r;
    Bool    val;

    scAutoUnlock    h1( rgnH );
    r = (HRgn*)*h1;

    val = r->fMaxBounds.Intersect( rec );

    return val;
}


/***************************************************************************/
/* EqualHRgn returns 1 if the regions have identical fSlivers, 0 o.w.   */

Bool EqualHRgn( const HRgnHandle    rgnA,
                const HRgnHandle    rgnB )
{
    const HRgn*     rA;
    const HRgn*     rB;
    const Sliver*   a;
    const Sliver*   b;
    const Sliver*   endA;
    MicroPoint      aX,
                    aY,
                    bX,
                    bY;
    Bool            same;

    scAutoUnlock    h1( rgnA );
    rA  = (const HRgn*)*h1;

    scAutoUnlock    h2( rgnB );
    rB  = (const HRgn*)*h2;

    if ( rA->fNumSlivers + rB->fNumSlivers == 0 )
        return true;

    if (    rA->fNumSlivers     != rB->fNumSlivers      ||
            rA->fVertInterval   != rB->fVertInterval    ||
            rA->fMaxBounds      != rB->fMaxBounds    ) {
        return false;
    }

    aX      = rA->fOrg.x;
    aY      = rA->fOrg.y;
    bX      = rB->fOrg.x;
    bY      = rB->fOrg.y;

    scAutoUnlock    h3( rA->fSlivers );
    a       = (const Sliver*)*h3;

    scAutoUnlock    h4( rB->fSlivers );
    b       = (const Sliver*)*h4;

    endA    = &a[rA->fNumSlivers];

    for ( same = true; same == true && a < endA; a++, b++ ) {
        throw;
    }

    DisposeHRgn( tempDstHRgn );
}


/***************************************************************************/
/* PtInHRgn returns 1 if the point is in the region, 0 o.w.  */

Bool PtInHRgn( const HRgnHandle rgnH,
               const scMuPoint& mPt )

{
    HRgn*       r;
    Sliver*     s;
    MicroPoint  mx,
                my;
    int         i;
    int         hi,
                low;
    MicroPoint  yPos;

    scAutoUnlock    h1( rgnH );
    r   = (HRgn*)*h1;

    mx  = mPt.x;
    my  = mPt.y;

    if ( !r->fMaxBounds.PinRect( mPt ) )
        return false;

    scAutoUnlock    h2( r->fSlivers );
    s   = (Sliver *)*h2;

    mx  -= r->fOrg.x;                /* cancel out effect of offsets */
    my  -= r->fOrg.y;

    low = 0;
    hi  = r->fNumSlivers - 1;
    for ( i = (hi+low) / 2; low <= hi; i = (hi+low) / 2 ) {
        if ( my < s[i].fSLTop)
            hi  = i - 1;
        else if ( my >= s[i].fSLTop + r->fVertInterval )
            low = i + 1;
        else
            break;
    }

    if ( low > hi )      /* didn't find any sliver with good y-value */
        return false;

    yPos    = s[i].fSLTop;
    if ( s[i].fSLx1 <= mx ) {
        for ( ;; i++ ) {
            if ( i >= r->fNumSlivers || s[i].fSLTop != yPos || mx < s[i].fSLx1 )
                return false;
            else if ( mx < s[i].fSLx2 )
                break;    /* success */
            else
                ;
        }
    }
    else {
        for ( ;; i-- ) {
            if ( i < 0 || s[i].fSLTop != yPos || s[i].fSLx2 <= mx )
                return false;
            else if ( s[i].fSLx1 <= mx )
                break;    /* success */
        }
    }

    return true;
```

```
                        // doing anything.
                        //
                    newA = newB = 0;
                    for ( ;; ) {
                        if ( newA ) {
                            if ( sA >= endA || sA->fSLTop + a->fOrg.y > yPos || sA->fSLx1 + a->fOrg.
x > sB->fSLx2 + b->fOrg.x) {
                                dst->AddSliver( left, sB->fSLx2 + b->fOrg.x, yPos );
                                sB++;
                                break;
                            }
                            else {
                                right = sA->fSLx1 + a->fOrg.x;
                            }
                        }
                        if ( newB ) {
                            if ( sB >= endB || sB->fSLTop + b->fOrg.y > yPos || sB->fSLx1 + b->fOrg.
x > sA->fSLx2 + a->fOrg.x) {
                                dst->AddSliver( left, sA->fSLx2 + a->fOrg.x, yPos );
                                sA++;
                                break;
                            }
                            else {
                                right = sB->fSLx1 + b->fOrg.x;
                            }
                        }

                        dst->AddSliver( left, right, yPos );

                        if ( sA->fSLx2 + a->fOrg.x == sB->fSLx2 + b->fOrg.x ) {
                            sA++;
                            sB++;
                            break;
                        }

                        if ( sA->fSLx2 + a->fOrg.x < sB->fSLx2 + b->fOrg.x ) {
                            newA = 1;
                            newB = 0;
                            left = sA->fSLx2 + a->fOrg.x;
                            sA++;
                        }
                        else {      /* sB->fSLx2 + b->fOrg.x < sA->fSLx2 + a->fOrg.x */
                            newB = 1;
                            newA = 0;
                            left = sB->fSLx2 + b->fOrg.x;
                            sB++;
                        }

                    }   /* inner for    */
                }       /* big else     */
            }           /* while        */

            for ( ; sA < endA; sA++ ) { /* sB >= endB */
                dst->AddSliver( sA->fSLx1 + a->fOrg.x, sA->fSLx2 + a->fOrg.x, sA->fSLTop + a->fOrg.y
 );
            }
            for ( ; sB < endB; sB++ ) { /* sA >= endA */
                dst->AddSliver( sB->fSLx1 + b->fOrg.x, sB->fSLx2 + b->fOrg.x, sB->fSLTop + b->fOrg.y
 );
            }

            scXRect arect( a->fOrigBounds );
            scXRect brect( a->fOrigBounds );
            arect.Union( brect );

            dst->fOrigBounds    = arect;

            dst->UpdateRealBounds( );
        }
        CopyHRgn( dstHRgn, tempDstHRgn );
    }
    catch ( ... ) {
        DisposeHRgn( tempDstHRgn );
    HRgn*               dst;
    const Sliver*       sA;
    const Sliver*       sB;
    const Sliver*       endA;
    const Sliver*       endB;
    MicroPoint          left,
                        right,
                        yPos;
    Bool                newA,
                        newB;

    raise_if ( srcHRgnA == 0 || srcHRgnB == 0 || dstHRgn == 0, scERRinput );

    try {
        scAutoUnlock    h1( srcHRgnA );
        a = (const HRgn*)*h1;

        scAutoUnlock    h2( srcHRgnB );
        b = (const HRgn*)*h2;

        raise_if ( a->fVertInterval != b->fVertInterval, scERRinput );

        tempDstHRgn = NewHRgn( b->fVertInterval );

        scAutoUnlock    h3( tempDstHRgn );
        dst = (HRgn*)*h3;

        dst->fVertInterval = a->fVertInterval;
        {
            scAutoUnlock    h4( a->fSlivers );
            sA      = (Sliver *)*h4;

            scAutoUnlock    h5( b->fSlivers );
            sB      = (Sliver *)*h5;

            endA    = &sA[a->fNumSlivers];
            endB    = &sB[b->fNumSlivers];

            while ( sA < endA && sB < endB ) {
                if ( sA->fSLTop + a->fOrg.y < sB->fSLTop + b->fOrg.y ) {
                    dst->AddSliver( sA->fSLx1 + a->fOrg.x, sA->fSLx2 + a->fOrg.x, sA->fSLTop + a->fO
rg.y );
                    sA++;
                }
                else if ( sB->fSLTop + b->fOrg.y < sA->fSLTop + a->fOrg.y ) {
                    dst->AddSliver( sB->fSLx1 + b->fOrg.x, sB->fSLx2 + b->fOrg.x, sB->fSLTop + b->fO
rg.y );
                    sB++;
                }              /* we know y positions are equal */
                else if ( sA->fSLx2 + a->fOrg.x <= sB->fSLx1 + b->fOrg.x ) {
                    dst->AddSliver( sA->fSLx1 + a->fOrg.x, sA->fSLx2 + a->fOrg.x, sA->fSLTop + a->fO
rg.y );
                    sA++;
                }
                else if ( sB->fSLx2 + b->fOrg.x <= sA->fSLx1 + a->fOrg.x ) {
                    dst->AddSliver( sB->fSLx1 + b->fOrg.x, sB->fSLx2 + b->fOrg.x, sB->fSLTop + b->fO
rg.y );
                    sB++;
                }
                else {      /* fSlivers intersect */
                    yPos = sA->fSLTop + a->fOrg.y;
                    if ( sA->fSLx1 + a->fOrg.x < sB->fSLx1 + b->fOrg.x ) {
                        left    = sA->fSLx1 + a->fOrg.x;
                        right   = sB->fSLx1 + b->fOrg.x;
                    }
                    else {
                        left    = sB->fSLx1 + b->fOrg.x;
                        right   = sA->fSLx1 + a->fOrg.x;
                    }

                        // the third case -- where the left edges are equal
                        // is handled implicitly. If AddSliver() sees x1
                        // and x2 which are equal, it will return without
        scAutoUnlock    h3( tempDstHRgn );
        dst = (HRgn *)*h3;

        dst->fVertInterval = a->fVertInterval;

        {
            scAutoUnlock    h4( a->fSlivers );
            sA      = (Sliver *)*h4;

            scAutoUnlock    h5( b->fSlivers );
            sB      = (Sliver *)*h5;

            endA    = &sA[a->fNumSlivers];
            endB    = &sB[b->fNumSlivers];

            for ( ; sA < endA; sA++ ) {
                yPos    = sA->fSLTop + a->fOrg.y;
                left    = sA->fSLx1 + a->fOrg.x;
                right   = sA->fSLx2 + a->fOrg.x;
                for ( ; sB < endB && sB->fSLTop + b->fOrg.y < yPos; sB++ )
                    ;
                for ( ; sB < endB && sB->fSLTop + b->fOrg.y == yPos && sB->fSLx2 + b->fOrg.x <= left
; sB++ )
                    ;

                                        /* process this sA sliver   */
                for ( flag = 1; flag; ) {   /* we may have to go through    */
                                        /* several sB fSlivers.     */
                    if ( sB >= endB || sB->fSLTop + b->fOrg.y > yPos || sB->fSLx1 + b->fOrg.x >= rig
ht ) {
                        flag    = 0;
                        dst->AddSliver( left, right, yPos );
                    }
                    else {
                        if ( sB->fSLx1 + b->fOrg.x > left ) {
                            dst->AddSliver( left, sB->fSLx1 + b->fOrg.x, yPos );
                        }
                        if ( sB->fSLx2 + b->fOrg.x >= right )
                            flag = 0;
                        else
                            left = sB++->fSLx2 + b->fOrg.x;
                    }
                }
            }

            dst->fOrigBounds    = a->fOrigBounds;

            dst->UpdateRealBounds( );
        }
        CopyHRgn( dstHRgn, tempDstHRgn );
    }
    catch ( ... ) {
        DisposeHRgn( tempDstHRgn );
        throw;
    }

    DisposeHRgn( tempDstHRgn );
}

/****************************************************************************/
/* Xor srcHRgnB and srcHRgnA and place result in third region.             */
/* If the regions are equal, the destination is set to the empty region.   */
/* The destination region may be one of the source regions.                */

void XorHRgn( const HRgnHandle  srcHRgnA,
              const HRgnHandle  srcHRgnB,
              HRgnHandle        dstHRgn )
{
    volatile HRgnHandle tempDstHRgn;
    const HRgn*         a;
    const HRgn*         b;
```

```
                                    right = sB->fSLx2 + b->fOrg.x;
                                sB++;
                            }
                            else
                                break;
                        }

                        dst->AddSliver( left, right, yPos );
                    }
                }

                for ( ; sA < endA; sA++ ) {      /* sB >= endB */
                    dst->AddSliver( sA->fSLx1 + a->fOrg.x, sA->fSLx2 + a->fOrg.x, sA->fSLTop + a->fOrg.y
 );
                }
                for ( ; sB < endB; sB++ ) {      /* sA >= endA */
                    dst->AddSliver( sB->fSLx1 + b->fOrg.x, sB->fSLx2 + b->fOrg.x, sB->fSLTop + b->fOrg.y
 );
                }

                scXRect arect( a->fOrigBounds );
                scXRect brect( b->fOrigBounds );
                arect.Union( brect );

                dst->fOrigBounds    = arect;

                dst->UpdateRealBounds( );
            }
            CopyHRgn( dstHRgn, tempDstHRgn );
        }
        catch ( ... ) {
            DisposeHRgn( tempDstHRgn );
            throw;
        }

        DisposeHRgn( tempDstHRgn );


/***********************************************************************/
/* Subtract srcHRgnB from srcHRgnA and place result in third region.     */
/* If the first region is empty, the destination is set to the empty region.*/
/* The destination region may be one of the source regions.              */

void DiffHRgn( const HRgnHandle srcHRgnA,
               const HRgnHandle srcHRgnB,
               HRgnHandle       dstHRgn )

        volatile HRgnHandle tempDstHRgn;
        const HRgn*         a;
        const HRgn*         b;
        HRgn*               dst;
        const Sliver*       sA;
        const Sliver*       sB;
        const Sliver*       endA;
        const Sliver*       endB;
        MicroPoint          left,
                            right,
                            yPos;
        int                 flag;

        raise_if ( srcHRgnA == 0 || srcHRgnB == 0 || dstHRgn == 0, scERRinput );

        try {
            scAutoUnlock    h1( srcHRgnA );
            a = (HRgn *)*h1;

            scAutoUnlock    h2( srcHRgnB );
            b = (HRgn *)*h2;

            raise_if ( a->fVertInterval != b->fVertInterval, scERRinput );

            tempDstHRgn = NewHRgn( b->fVertInterval );
```

```
    const Sliver*       endB;
    MicroPoint          left,
                        right,
                        yPos;

    raise_if ( srcHRgnA == 0 || srcHRgnB == 0 || dstHRgn == 0, scERRinput );

    try {
        scAutoUnlock    h1( srcHRgnA );
        a = (const HRgn*)*h1;

        scAutoUnlock    h2( srcHRgnB );
        b = (const HRgn *)*h2;

        raise_if ( a->fVertInterval != b->fVertInterval, scERRinput );

        tempDstHRgn = NewHRgn( b->fVertInterval );

        scAutoUnlock    h3( tempDstHRgn );
        dst = (HRgn*)*h3;

        dst->fVertInterval = a->fVertInterval;

        {
            scAutoUnlock    h4( a->fSlivers );
            sA      = (const Sliver *)*h4;

            scAutoUnlock    h5( b->fSlivers );
            sB      = (const Sliver*)*h5;

            endA    = &sA[a->fNumSlivers];
            endB    = &sB[b->fNumSlivers];

            while ( sA < endA && sB < endB ) {
                if ( sA->fSLTop + a->fOrg.y < sB->fSLTop + b->fOrg.y ) {
                    dst->AddSliver( sA->fSLx1 + a->fOrg.x, sA->fSLx2 + a->fOrg.x, sA->fSLTop + a->fO
rg.y );
                    sA++;
                }
                else if ( sB->fSLTop + b->fOrg.y < sA->fSLTop + a->fOrg.y ) {
                    dst->AddSliver( sB->fSLx1 + b->fOrg.x, sB->fSLx2 + b->fOrg.x, sB->fSLTop + b->fO
rg.y );
                    sB++;
                }               /* we know y positions are equal */
                else if ( sA->fSLx2 + a->fOrg.x < sB->fSLx1 + b->fOrg.x ) {
                    dst->AddSliver( sA->fSLx1 + a->fOrg.x, sA->fSLx2 + a->fOrg.x, sA->fSLTop + a->fO
rg.y );
                    sA++;
                }
                else if ( sB->fSLx2 + b->fOrg.x < sA->fSLx1 + a->fOrg.x ) {
                    dst->AddSliver( sB->fSLx1 + b->fOrg.x, sB->fSLx2 + b->fOrg.x, sB->fSLTop + b->fO
rg.y );
                    sB++;
                }
                else {          /* build a sliver until change y-coord */
                    /* or find disjoint sliver */
                    /* or A's or B's fSlivers end */
                    yPos    = sA->fSLTop + a->fOrg.y;
                    left    = MIN( sA->fSLx1 + a->fOrg.x, sB->fSLx1 + b->fOrg.x );
                    right   = MAX( sA->fSLx2 + a->fOrg.x, sB->fSLx2 + b->fOrg.x );
                    sA++;
                    sB++;

                    for ( ;; ) {
                        if ( sA < endA && sA->fSLTop + a->fOrg.y == yPos && sA->fSLx1 + a->fOrg.x <=
 right ) {
                            if ( sA->fSLx2 + a->fOrg.x > right )
                                right = sA->fSLx2 + a->fOrg.x;
                            sA++;
                        }
                        else if ( sB < endB && sB->fSLTop + b->fOrg.y == yPos && sB->fSLx1 + b->fOrg
.x <= right ) {
                            if ( sB->fSLx2 + b->fOrg.x > right )
                b->fNumSlivers == 0                 ||
                a->fMaxBounds.x1 >= b->fMaxBounds.x2    ||
                b->fMaxBounds.x1 >= a->fMaxBounds.x2    ||
                a->fMaxBounds.y1 >= b->fMaxBounds.y2    ||
                b->fMaxBounds.y1 >= a->fMaxBounds.y2    ) {
            ;;
        }
        else {
            scAutoUnlock    h4( a->fSlivers );
            sA      = (const Sliver*)*h4;

            scAutoUnlock    h5( b->fSlivers );
            sB      = (const Sliver*)*h5;

            endA    = &sA[a->fNumSlivers];
            endB    = &sB[b->fNumSlivers];

            while ( sA < endA && sB < endB ) {
                if ( sA->fSLTop + a->fOrg.y < sB->fSLTop + b->fOrg.y )
                    sA++;
                else if ( sB->fSLTop + b->fOrg.y < sA->fSLTop + a->fOrg.y )
                    sB++;
                else if ( sA->fSLx2 + a->fOrg.x <= sB->fSLx1 + b->fOrg.x )
                    sA++;
                else if ( sB->fSLx2 + b->fOrg.x <= sA->fSLx1 + a->fOrg.x )
                    sB++;
                else {              /* fSlivers intersect */
                    dst->AddSliver( MAX( sA->fSLx1 + a->fOrg.x, sB->fSLx1 + b->fOrg.x ),
                                    MIN( sA->fSLx2 + a->fOrg.x, sB->fSLx2 + b->fOrg.x ), sA->fSLTop
 a->fOrg.y );
                    if ( sA->fSLx2 + a->fOrg.x < sB->fSLx2 + b->fOrg.x )
                        sA++;
                    else
                        sB++;
                }
            }

            scXRect arect( a->fOrigBounds );
            scXRect brect( b->fOrigBounds );
            arect.Intersect( brect );

            dst->fOrigBounds    = arect;

            dst->UpdateRealBounds( );
        }
        CopyHRgn( dstHRgn, tempDstHRgn );
    }

    catch ( ... ) {
        DisposeHRgn( tempDstHRgn );
        throw;
    }

    DisposeHRgn( tempDstHRgn );
}


/*****************************************************************************/
/* Calculate the union of the two regions and place result  in third region */
/* If both regions are empty, the destination is set to the empty region.   */
/* The destination region may be one of the source regions.                 */

void UnionHRgn( const HRgnHandle    srcHRgnA,
                const HRgnHandle    srcHRgnB,
                HRgnHandle  dstHRgn )
{
    volatile HRgnHandle tempDstHRgn;
    const HRgn*         a;
    const HRgn*         b;
    HRgn*               dst;
    const Sliver*       sA;
    const Sliver*       sB;
    const Sliver*       endA;
```

```
{
    Sliver* s;
    int     i;

    if( x2 <= x1 )
        return;

    if ( fMaxSlivers <= fNumSlivers ) {
        fSlivers = MEMResizeHnd( fSlivers, MEMGetSizeHnd( fSlivers ) +  sliver_growSize * sizeof( Sl
iver ) );
        fMaxSlivers += sliver_growSize;
    }

    UpdateBounds( x1, x2, y );

    x1  -= fOrg.x;
    x2  -= fOrg.y;
    y   -= fOrg.y;


    scAutoUnlock    h1( fSlivers );
    s       = (Sliver *)*h1;
    i       = fNumSlivers - 1;
    for ( ; 0 <= i && ( s[i].fSLTop > y || s[i].fSLTop == y && s[i].fSLx1 > x1 ); i-- )
        ;
    i++;

    if ( i < fNumSlivers )
        SCmemmove( &s[i+1], &s[i], (long)( fNumSlivers - i ) * sizeof( Sliver ) );

    (&s[i])->SetSliver( x1, x2, y );
    fNumSlivers++;
}

/**************************************************************************/
/* Calculate the intersection of the two regions and place result        */
/* in third region. If the regions have no intersection, or one          */
/* of the regions is empty, the destination is set to the empty region.  */
/* The destination region may be one of the source regions.              */

void SectHRgn( const HRgnHandle srcHRgnA,
               const HRgnHandle srcHRgnB,
               HRgnHandle       dstHRgn )
{
    volatile HRgnHandle tempDstHRgn;
    const HRgn*         a;
    const HRgn*         b;
    HRgn*               dst;
    const Sliver*       sA;
    const Sliver*       sB;
    const Sliver*       endA;
    const Sliver*       endB;

    raise_if ( srcHRgnA == 0 || srcHRgnB == 0 || dstHRgn == 0, scERRinput );

    try {
        scAutoUnlock    h1( srcHRgnA );
        a = (const HRgn*)*h1;

        scAutoUnlock    h2(srcHRgnB );
        b = (const HRgn*)*h2;

        raise_if ( a->fVertInterval != b->fVertInterval, scERRinput );

        tempDstHRgn = NewHRgn( b->fVertInterval );

        scAutoUnlock    h3( tempDstHRgn );
        dst = (HRgn *)*h3;

        dst->fVertInterval = a->fVertInterval;

        if (    a->fNumSlivers == 0                         ||
void HRgn::UpdateBounds( MicroPoint x1,
                         MicroPoint x2,
                         MicroPoint y )
{
    if ( fNumSlivers == 0 )
        fMaxBounds.Set( x1, y, x2, y + fVertInterval );
    else {
#if 1
        scXRect urect( x1, y, x2, y + fVertInterval );
        fMaxBounds.Union( urect );
#else
        MicroPoint diff = x2 - r->fMaxBounds.x1 - r->fMaxBounds.Width();
        if ( 0L < diff )
            r->fMaxBounds.x2 += diff;


        diff = r->fMaxBounds.x1 - x1;
        if ( 0L < diff ) {
            r->fMaxBounds.x1 -= diff;
            r->fMaxBounds.x2 += diff;
        }


        diff = y + r->fVertInterval - r->fMaxBounds.y1 - r->fMaxBounds.Depth();
        if ( 0L < diff )
            r->fMaxBounds.y2 += diff;

        diff = r->fMaxBounds.y1 - y;
        if ( 0L < diff ) {
            r->fMaxBounds.y1 -= diff;
            r->fMaxBounds.y2 += diff;
        }
#endif
    }
}

/************************************************************************/
/* Make sure the original bounds have not gotten out of sync with the  */
/* current shape of the region.                                         */

void HRgn::UpdateRealBounds( )
{
    MicroPoint  left    = fOrigBounds.x1;
    MicroPoint  top     = fOrigBounds.y1;
    MicroPoint  right   = fOrigBounds.x2;
    MicroPoint  bottom  = fOrigBounds.y2;
    MicroPoint  slvSize = fVertInterval;

    if ( left != fMaxBounds.x1 )
        left = fMaxBounds.x1;

    if ( top < fMaxBounds.y1 || top + slvSize > fMaxBounds.y1 )
        top = fMaxBounds.y1;

    if ( right != fMaxBounds.x2 )
        right = fMaxBounds.x2;

    if ( bottom > fMaxBounds.y2 || bottom + slvSize < fMaxBounds.y2 )
        bottom = fMaxBounds.y2;

    fOrigBounds.Set( left, top, right, bottom );
}

/************************************************************************/
/* Update the region by adding the given sliver.          */
/* If horizontal coordinates are not possible, AddSliver  */
/* takes no action, and returns scSuccess.            */

void HRgn::AddSliver( MicroPoint    x1,
                      MicroPoint    x2,
                      MicroPoint    y )
        newMax          = r->fNumSlivers + sliver_growSize - ( r->fNumSlivers % sliver_growSize );
        r->fSlivers     = MEMResizeHnd( r->fSlivers, newMax * sizeof( Sliver ) );
        r->fMaxSlivers  = newMax;
    }

    scAutoUnlock    h2( r->fSlivers );
    sliverPtr   = (Sliver*)*h2;

    endSliver   = &sliverPtr[r->fNumSlivers];
    for ( ; sliverPtr < endSliver; top += r->fVertInterval, sliverPtr++ )
        sliverPtr->SetSliver( left, right, top );

}


/****************************************************************************/
/* Move the region, unchanged, a distance of dx horizontally and          */
/* dy vertically on the coordinate plane.                                  */

void TranslateHRgn( HRgnHandle  rgnH,
                    MicroPoint  dx,
                    MicroPoint  dy )
{
    scAutoUnlock    h1( rgnH );
    HRgn*   r = (HRgn*)*h1;

    dy = PointMassage( dy, r->fVertInterval );

    r->fOrg.Translate( dx, dy );
    r->fMaxBounds.Translate( dx, dy );
    r->fOrigBounds.Translate( dx, dy );
}


/****************************************************************************/
/* Redo the bounds of the region by running through the fSlivers.    */

void HRgn::SetBounds( )
{
    Sliver*     s;
    Sliver*     end;
    MicroPoint  left,
                right,
                top,
                bottom;

    if ( fNumSlivers == 0 ) {
        fMaxBounds.Set( 0, 0, 0, 0 );
        return;
    }

    scAutoUnlock    h1( fSlivers );
    s   = (Sliver*)*h1;
    end = &s[fNumSlivers];

    left    = s->fSLx1;
    right   = s->fSLx2;
    top     = bottom    = s->fSLTop;

    for ( s++; s < end; s++ ) {
        left    = MIN( s->fSLx1, left );
        right   = MAX( s->fSLx2, right );
        bottom  = MAX( s->fSLTop, bottom );
    }

    fMaxBounds.Set( left, top, right, bottom + fVertInterval );
    fMaxBounds.Translate( fOrg );
}


/****************************************************************************/
/* Update the bounds of the region with the new sliver coordinates. */
```

```
    dst->fMaxBounds     = src->fMaxBounds;

    dst->fOrg           = src->fOrg;

    dst->fVertInterval  = src->fVertInterval;
    dst->fNumSlivers    = src->fNumSlivers;


    scAutoUnlock    h3( dst->fSlivers );
    to      = (Sliver*)*h3;

    scAutoUnlock    h4( src->fSlivers );
    from    = (Sliver*)*h4;

    SCmemmove( to, from, src->fNumSlivers * sizeof( Sliver ) );
}


/************************************************************************/
/* Destroy previous structure of region and set it to the rectangle */
/* defined by (0,0) (0,0).                                          */

void SetEmptyHRgn( HRgnHandle   rgnH )
{
    HRgn*   r;

    scAutoUnlock    h( rgnH );
    r = (HRgn*)*h;

    r->fSlivers = MEMResizeHnd( r->fSlivers, sliver_growSize * sizeof( Sliver ) );

    r->fOrigBounds.Set( 0, 0, 0, 0 );
    r->fMaxBounds.Set( 0, 0, 0, 0 );

    r->fOrg.Set( 0, 0 );

    r->fNumSlivers      = 0;
    r->fMaxSlivers      = sliver_growSize;
}


/************************************************************************/
/* Destroy previous structure of region and set it to the rectangle */
/* defined by rec.                                                  */

void RectHRgn( HRgnHandle       rgnH,
               const scXRect&   rec )
{
    HRgn*       r;
    Sliver*     sliverPtr;
    Sliver*     endSliver;
    MicroPoint  left;
    MicroPoint  right;
    MicroPoint  top;
    int         newMax;

    scAutoUnlock    h1( rgnH );
    r = (HRgn*)*h1;

    r->fOrigBounds      = rec;

    r->fMaxBounds.x1    = left      = rec.x1;
    r->fMaxBounds.y1    = top       = Massage( rec.y1, r->fVertInterval, -1 );
    r->fMaxBounds.x2    = right     = rec.x2;
    r->fMaxBounds.y2    = Massage( rec.y2, r->fVertInterval, 1 );

    r->fOrg.x       = 0;
    r->fOrg.y       = 0;

        /* guaranteed no remainder */
    r->fNumSlivers  = (int)( r->fMaxBounds.Depth() / r->fVertInterval );

    if ( r->fMaxSlivers < r->fNumSlivers ) {
```

```
{
#endif

    HRgnHandle  rgnH;
    HRgn*       r;

#if defined( MEM_DEBUG )
    rgnH = (HRgnHandle)MEMAllocHndDebug( sizeof( HRgn ), fn, line );
#else
    rgnH = (HRgnHandle)MEMAllocHnd( sizeof( HRgn ) );
#endif

    scAutoUnlock    h( rgnH );
    r = (HRgn*)*h;

    r->fOrigBounds.Set( 0, 0, 0, 0 );
    r->fMaxBounds.Set( 0, 0, 0, 0 );

    r->fOrg.Set( 0, 0 );

    r->fVertInterval    = theSliverSize;
    r->fNumSlivers      = 0;
    r->fMaxSlivers      = sliver_growSize;

#if defined( MEM_DEBUG )
    r->fSlivers = MEMAllocHndDebug( sliver_growSize * sizeof( Sliver ), fn, line );
#else
    r->fSlivers = MEMAllocHnd( sliver_growSize * sizeof( Sliver ) );
#endif

    return rgnH;
}

/*************************************************************************/
/* dispose of a region */

void DisposeHRgn( HRgnHandle rgnH )
{
    HRgn* r = (HRgn*)scMemDeref( rgnH );

    MEMFreeHnd( r->fSlivers );

    MEMFreeHnd( rgnH );
}

/*************************************************************************/
/* copy source region to destination region. Space must already be */
/* allocated for destination region.                                 */

void CopyHRgn( HRgnHandle       dstRgn,
               const HRgnHandle srcRgn )
{
    HRgn*           dst;
    Sliver*         to;
    const HRgn*     src;
    const Sliver*   from;
    int             diff;

    raise_if ( ( dstRgn == 0 ) || ( srcRgn == 0 ), scERRinput );

    scAutoUnlock    h1( srcRgn );
    src = (HRgn*)*h1;

    scAutoUnlock    h2( dstRgn );
    dst = (HRgn*)*h2;

    diff            = src->fMaxSlivers - dst->fMaxSlivers;
    dst->fSlivers   = MEMResizeHnd( dst->fSlivers, (ulong)src->fMaxSlivers * sizeof(Sliver) );

    dst->fMaxSlivers += diff;

    dst->fOrigBounds    = src->fOrigBounds;
```

```
/* dealing with the top edge; if it is 1, the bottom edge.                  */
/* Err on the side of making region larger.                                 */

typedef enum eEdgeModes {
    eTopEdge,
    eBottomEdge
} eEdgeMode;

static MicroPoint Massage ( MicroPoint  pos,
                            MicroPoint  size,
                            int     mode )
{
    MicroPoint      rem;

    if ( ( rem = ABS( pos % size ) ) != 0 ) {
        /* take abs because sign of result of % is machine dependent. */
        if ( pos < 0 ) {                        /* negative y-coordinate */
            if ( mode < 0 )                     /* top edge */
                return( pos - ( size - rem ) );
            else                                /* bottom edge */
                return( pos + rem );
        }
        else {                                  /* positive y-coordinate */
            if ( mode < 0 )                     /* top edge */
                return( pos - rem );
            else                                /* bottom edge */
                return( pos + size - rem );
        }
    }
    else                                        /* no remainder */
        return pos;
}

/**************************************************************************/
/* Like Massage, but simpler, since it is used for offset points,   */
/* and we don't have to worry about which edge we are looking at.   */

static MicroPoint PointMassage( MicroPoint  ypt,
                                MicroPoint  size )
{
    MicroPoint  rem;

        /* take abs because sign of result of % is machine dependent. */
    if ( ypt ) {
        rem = ABS( ypt % size );

        if ( rem <= ( size/2 ) ) {
            if ( ypt < 0 )
                ypt += rem;
            else
                ypt -= rem;
        }
        else {
            if ( ypt < 0)
                ypt -= size - rem;
            else
                ypt += size - rem;
        }
    }
    return ypt;
}


/**************************************************************************/
/* Allocate space for a region and initialize everything to zero.  */

#if SCDEBUG > 1
HRgnHandle NewHRgnDebug( MicroPoint     theSliverSize,
                         const char*    fn,
                         int            line )
{
#else
HRgnHandle NewHRgn( MicroPoint  theSliverSize )
```

```
/****************************************************************************

    File:       SCREGION.C


    $Header: /Projects/Toolbox/ct/SCREGION.CPP 3     5/30/97 8:45a Wmanis $

    Contains:   HiRes region implementation.

    Written by: Lucas

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

****************************************************************************/

#include "scbezier.h"
#include "scmem.h"
#include "scfileio.h"
#include "scregion.h"
#include <limits.h>

#define edge_growSize           64


/****************************************************************************/
/***********************  REGION  *******************************************/
/****************************************************************************/


struct HEdge {                      /* used in scan line conversion         */
    MicroPoint  y1;                 /* y-coord of first vertex              */
    MicroPoint  y2;                 /* y-coord of second vertex             */
    MicroPoint  x1;                 /* x-coord of first vertex              */
    MicroPoint  x;                  /* current x-coord (moving along edge)  */
    MicroPoint  dx;
    MicroPoint  dy;

};

typedef scMemHandle HEdgeHandle;


/*********************************************************************/
/*********************************************************************/
/* LOCAL PROTOTYPES */

static MicroPoint   Massage( MicroPoint, MicroPoint, int );

static void         PlainInsetHRgn( HRgnHandle, MicroPoint, MicroPoint );
static void         XGrowHRgn( HRgnHandle, MicroPoint );
static void         XShrinkHRgn( HRgnHandle, MicroPoint );
static void         YGrowHRgn( HRgnHandle, MicroPoint );
static void         YShrinkHRgn( HRgnHandle, MicroPoint );

static void         ConstructEdge( const scVertex*,
                                   HEdgeHandle*,
                                   int*,
                                   MicroPoint* );

static int          RemInactive( HEdge*, MicroPoint, int );


/****************************************************************************/
/* Relocate top or bottom edge of an enclosing rectangle so that its       */
/* y-position is evenly divisible by sliver_size. If mode is -1, we are     */
```

```
    int         fMaxSlivers;    // space exists for this many slivers

    scMemHandle fSlivers;
};

/*******************************************************************/
/*******************************************************************/
/* PROTOTYPES */

#if SCDEBUG > 1
    HRgnHandle  NewHRgnDebug( MicroPoint, const char*, int );
    #define NewHRgn( mp )       NewHRgnDebug( (mp), __FILE__, __LINE__ )
#else
    HRgnHandle  NewHRgn( MicroPoint );
#endif

void        DisposeHRgn( HRgnHandle );
Bool        EmptyHRgn( HRgnHandle );
Bool        EqualHRgn( const HRgnHandle, const HRgnHandle );
Bool        PtInHRgn( const HRgnHandle, const scMuPoint& );
void        RectHRgn( HRgnHandle, const scXRect& );
void        PolyHRgn( HRgnHandle, const scVertex* );
void        CopyHRgn( HRgnHandle, const HRgnHandle );
void        SectHRgn( const HRgnHandle, const HRgnHandle, HRgnHandle );
void        UnionHRgn( const HRgnHandle, const HRgnHandle, HRgnHandle );
void        DiffHRgn( const HRgnHandle, const HRgnHandle, HRgnHandle );
void        XorHRgn( const HRgnHandle, const HRgnHandle, HRgnHandle );
void        TranslateHRgn( HRgnHandle, MicroPoint, MicroPoint );
void        InsetHRgn( HRgnHandle, MicroPoint, MicroPoint, Bool );
void        SetEmptyHRgn( HRgnHandle );
MicroPoint  RGNSliverSize( const HRgnHandle );
Bool        RectInHRgn ( const HRgnHandle, const scXRect& );


long        RGNExternalSize( const HRgnHandle, long );
void        RGNGetExtents( const HRgnHandle, scXRect& );
MicroPoint  RGNMaxDepth( const HRgnHandle );
Bool        RGNtoFile( APPCtxPtr, IOFuncPtr, HRgnHandle, int );
HRgnHandle  RGNfromFile( APPCtxPtr, IOFuncPtr, int );



#if SCDEBUG > 0
void        RGNInvertSlivers( const HRgnHandle,
                              APPDrwCtx,
                              HiliteFuncPtr,
                              const scSize&,
                              int vertflow );
#endif


#endif
```

```
/****************************************************************************

    File:       SCREGION.H


    $Header: /Projects/Toolbox/ct/SCREGION.H 3     5/30/97 8:45a Wmanis $

    Contains:   HiRes region definition.

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

****************************************************************************/

#ifndef _H_SCREGION
#define _H_SCREGION

#include "sctypes.h"


#define sliver_growSize     64

#define HRgnSize( n )       ((long)(sizeof(HRgn)+((n)*sizeof(Sliver))))

struct Sliver {
    MicroPoint  fSLx1,
                fSLx2;      // horizontal extants of sliver
    MicroPoint  fSLTop;     // top of sliver


    void        SetSliver( MicroPoint a, MicroPoint b, MicroPoint c )
                    { fSLx1 = a, fSLx2 = b, fSLTop = c; }
};

struct HEdge;

class HRgn {
public:
    void        SetBounds( void );
    void        UpdateBounds( MicroPoint, MicroPoint, MicroPoint );
    void        UpdateRealBounds( void );
    void        AddSliver( MicroPoint, MicroPoint, MicroPoint );
    Bool        IsBorder( int, int );

    void        CorpSliver( Sliver**, MicroPoint, MicroPoint, MicroPoint );

    void        InsertSliver( MicroPoint, MicroPoint, MicroPoint,
                              Sliver**, int, int );

    void        ScanEdges( HEdge*, long, int );


    MicroPoint  FirstLinePos( MicroPoint    firstLinePos,
                              MicroPoint    leading );

    void        SectRect( scXRect&, MicroPoint, MicroPoint, MicroPoint);

    long        fVersion;
    scXRect     fOrigBounds;    // the original bounds of the region
    scXRect     fMaxBounds;     // max bounds of region
    scMuPoint   fOrg;           // locations of slivers are with
                                // respect to this point
    MicroPoint  fVertInterval;  // vertical size of slivers
    int         fNumSlivers;
```

```
/* ==================================================================== */

void scRubiData::TransOffsets( long offset )
{
    fStartOffset    += offset;
    fEndOffset      += offset;
}

/* ==================================================================== */
```

```
scRubiData::scRubiData( const UCS2 *ch, long start, long end, TypeSpec ts )
{
    int len = MIN( CharacterBufLen( ch ), 16 );

    memcpy( fCh, ch, len * sizeof( UCS2 ) );
    fCh[len] = 0;
    fStartOffset        = start;
    fEndOffset          = end;
    fRubiSpec           = ts;
    fOrg.Set( 0, 0 );
    fExtents.Set( 0, 0, 0, 0 );
}

/* ==================================================================== */

void scRubiData::Read( APPCtxPtr    ctxPtr,
                       IOFuncPtr    readFunc )
{
    char buf[kRubiDataSize2];

    int readin = (*readFunc)( ctxPtr, buf, 4 );

        // this nonsense is to fix a bug int the original i/o and to
        // try and maintain fill compatability
    if ( *(long*)buf == kRubiMagic ) {
        int readin = (*readFunc)( ctxPtr, buf + 4, sizeof( buf ) - 4 );

        ::memcpy( fCh, buf + 4, 16 * sizeof( UCS2 ) );
        fCh[16]         = 0;
        fStartOffset    = *(long *)( buf + 36 );
        fEndOffset      = *(long *)( buf + 40 );
        fRubiSpec       = (TypeSpec)*(long *)( buf + 44 );
        long diskid = APPPointerToDiskID( ctxPtr,
                                                (*this)[i].spec().ptr(),
                                                diskidTypespec );

    }
    else {
        int readin = (*readFunc)( ctxPtr, buf + 4, kRubiDataSize - 4 );

        ::memcpy( fCh, buf, 8 * sizeof( UCS2 ) );
        fCh[8]          = 0;
        fStartOffset    = *(long *)( buf + 16 );
        fEndOffset      = *(long *)( buf + 20 );
        fRubiSpec       = (TypeSpec)*(long *)( buf + 24 );
    }

/* ==================================================================== */

void scRubiData::PtrRestore( void )
{
//  fRubiSpec = (TypeSpec)APPDiskIDToPointer( (ulong)fRubiSpec );
}

/* ==================================================================== */

void scRubiData::Write( APPCtxPtr   ctxPtr,
                        IOFuncPtr       writeFunc )
{
    char buf[kRubiDataSize2];

    *(long *)buf            = kRubiMagic;

    memcpy( buf + 4, fCh, 16 * sizeof( UCS2 ) );

    *(long *)( buf + 36 )   = fStartOffset;
    *(long *)( buf + 40 )   = fEndOffset;
//  *(long *)( buf + 44 )   = (ulong)APPPointerToDiskID( fRubiSpec );

    int written = (*writeFunc)( ctxPtr, buf, sizeof( buf ) );
}
```

```
void scRubiArray::DeleteRubiData( long offset )
{
    int         index;
    scRubiData  rd;

    index = FirstSuccess( is_rubi_at, offset, offset );

    if ( index >= 0 )
        RemoveDataAt( index );
}

/* ==================================================================== */
// delete rubi data between the indicated offsets

void scRubiArray::DeleteRubiData( long start, long end )
{
    int index;

    while ( ( index = FirstSuccess( is_rubi_at, start, end ) ) >= 0 ) {
        RemoveDataAt( index );
    }

    BumpRubiData( end, start - end );
}

/* ==================================================================== */

    static void readrubidata( ElementPtr ptr, long ctxPtr, long readFunc )
    {
        ((scRubiData *)ptr)->Read( (APPCtxPtr)ctxPtr, (IOFuncPtr)readFunc );
    }

void scRubiArray::Read( APPCtxPtr   ctxPtr,
                        IOFuncPtr       readFunc,
                        int             numToRead )
{
    GrowSlots( numToRead );
    fNumItems   = numToRead;

    DoForEach( readrubidata, (long)ctxPtr, (long)readFunc );
}

/* ==================================================================== */

    static void writerubidata( ElementPtr ptr, long ctxPtr, long writeFunc )
    {
        ((scRubiData *)ptr)->Write( (APPCtxPtr)ctxPtr, (IOFuncPtr)writeFunc );
    }

void scRubiArray::Write( APPCtxPtr  ctxPtr,
                         IOFuncPtr      writeFunc )
{
    DoForEach( writerubidata, (long)ctxPtr, (long)writeFunc );
}

/* ==================================================================== */

    static void ptrrestorerubidata( ElementPtr ptr )
    {
        ((scRubiData *)ptr)->PtrRestore();
    }


void scRubiArray::PtrRestore( )
{
    DoForEach( ptrrestorerubidata );
}

/* ==================================================================== */
/* ==================================================================== */
/* ==================================================================== */
/* ==================================================================== */
```

```
    extern "C" {
        static int scCDecl rubi_sort( const void *p1, const void *p2 )
        {
            scRubiData& rd1 = *(scRubiData *)p1;
            scRubiData& rd2 = *(scRubiData *)p2;

            return (int)(rd1.fStartOffset - rd2.fStartOffset) ;
        }
    }

// add rubi data and sort the data

Bool scRubiArray::AddRubiData( scRubiData& rd )
{
    if ( IsRubiData( rd.fStartOffset, rd.fEndOffset ) )
        return false;

    AppendData( (ElementPtr)&rd );
    QuickSort( rubi_sort );
    return true;
}

/* ==================================================================== */

Bool scRubiArray::GetRubiAt( scRubiData& rd, long offset )
{
    long        index;
    int         nth;

    for ( nth = 1; ( index = NthSuccess( is_rubi_at, nth, offset, offset ) ) >= 0; nth++ ) {
        GetDataAt( (int)index, (ElementPtr)&rd );
        if ( offset > rd.fStartOffset && offset < rd.fEndOffset )
            return true;
    }
    return false;
}

/* ==================================================================== */

Bool scRubiArray::GetNthRubi( int& index, scRubiData& rubiData, int nth, long start, long end )
{
    index = (int)NthSuccess( is_rubi_at, nth, start, end );

    if ( index < 0 )
        return false;

    GetDataAt( index, (ElementPtr)&rubiData );

    return true;
}

/* ==================================================================== */
// place the rubiarray into the existing rubi array at the
// indicated offset with it covering the number of chars
// indicated

void scRubiArray::Paste( const scRubiArray& ra, long offset, int size )
{
    scRubiData  rd;
    int         i;

    BumpRubiData( offset, size );

    for ( i = 0; i < ra.GetNumItems(); i++ ) {
        ra.GetDataAt( i, (ElementPtr)&rd );
        rd.TransOffsets( offset );
        AddRubiData( rd );
    }
}

/* ==================================================================== */
// delete rubi data at the offset
```

```
    return kRubiDataSize2 * GetNumItems();
}

/* ================================================================== */
// is there an annotation at this location

    static Bool is_rubi_at( const ElementPtr ptr, long start, long end )
    {
        scRubiData& rd = (scRubiData&)*ptr;
        scRange r1( rd.fStartOffset, rd.fEndOffset );
        scRange r2( start, end );

        return r1.Exclusive_Sect( r2 );
    }

Bool scRubiArray::IsRubiData( long start, long end )
{
    return FirstSuccess( is_rubi_at, start, end  ) >= 0;
}

/* ================================================================== */
// if the offset is at a border we will not return true

Bool scRubiArray::IsRubiData( long offset )
{
    long        index;
    int         nth;
    scRubiData  rd;

    for ( nth = 1; ( index = NthSuccess( is_rubi_at, nth, offset, offset ) ) >= 0; nth++ ) {
        GetDataAt( (int)index, (ElementPtr)&rd );
        if ( offset > rd.fStartOffset && offset < rd.fEndOffset )
            return true;
    }
    return false;
}

/* ================================================================== */
// as we edit text change the annotations

    static void bump_rubi_data( ElementPtr ptr, long start, long amount )
    {
        scRubiData& rd = (scRubiData&)*ptr;

        if ( start <= rd.fStartOffset )
            rd.TransOffsets( amount );
    }

void scRubiArray::BumpRubiData( long start, long amount )
{
    DoForEach( bump_rubi_data, (long)start, amount );
}

/* ================================================================== */
// apply the style to the rubidata found within the bounds

void scRubiArray::ApplyStyle( long start, long end, TypeSpec ts )
{
    scRubiData  rd;
    long        index;
    int         nth;

    for ( nth = 1; ( index = NthSuccess( is_rubi_at, nth, start, end ) ) >= 0; nth++ ) {
        GetDataAt( (int)index, (ElementPtr)&rd );
        if ( rd.fStartOffset >= start && rd.fStartOffset < end ) {
            rd.fRubiSpec = ts;
            AlterDataAt( (int)index, (ElementPtr)&rd );
        }
    }
}

/* ================================================================== */
```

```
/*==================================================================

    File:       crubi.c

    $Header: /Projects/Toolbox/ct/scrubi.cpp 2     5/30/97 8:45a Wmanis $

    Contains:   Impelmentation of rubi storarge.

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

==================================================================*/

#include "scrange.h"
#include "sccallbk.h"
#include "scrubi.h"
#include "scannota.h"
#include "sctbobj.h"

#define kRubiMagic      0x1a1a1a1a

    // for writing out rubi data - str + start + end + spec
#define kRubiDataSize   ( 16 + 4 + 4 + 4 )

    // for writing out rubi data - magic + str + start + end + spec
#define kRubiDataSize2  ( 4 + 32 + 4 + 4 + 4 )

/* ================================================================== */

void scAnnotation::Set( UCS2 *ch, int paraoffset, int start, int end  )
{
    memcpy( fCharStr, ch, 34 );
    fAnnotate       = fCharStr[0] != 0;
    fParaOffset     = paraoffset;
    fStartOffset    = start;
    fEndOffset      = end;
}

/* ================================================================== */


scRubiArray::scRubiArray( )
    : scMemArray( sizeof( scRubiData ) )
{
}

/* ================================================================== */

scRubiArray::~scRubiArray()
{
}

/* ================================================================== */

scRubiArray& scRubiArray::operator=( const scRubiArray& ra )
{
    scMemArray::operator=( ra );
    return *this;
}

/* ================================================================== */

long scRubiArray::ExternalSize( void )
{
    void     ApplyStyle( long start, long end, TypeSpec ts );

             // add rubi data
    Bool     AddRubiData( scRubiData& );

             // delete rubi data
    void     DeleteRubiData( long );

             // delete rubi data between the indicated offsets
    void     DeleteRubiData( long, long );

    scRubiArray&    operator=( const scRubiArray& );

    long     ExternalSize( void );
    void     Read( APPCtxPtr, IOFuncPtr, int numread );
    void     PtrRestore( void );
    void     Write( APPCtxPtr, IOFuncPtr );
};

#endif
```

```
/*================================================================

    File:      crubi.h

    $Header: /Projects/Toolbox/ct/scrubi.h 2     5/30/97 8:45a Wmanis $

    Contains:  rubi data

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

================================================================*/

#ifndef _H_CRUBI
#define _H_CRUBI

#include "scmemarr.h"


struct scRubiData {
    UCS2        fCh[18];            // the rubi characters, NULL terminated
    long        fStartOffset;       // start offset in stream
    long        fEndOffset;         // end offset in stream
    scMuPoint   fOrg;               // drawing origin
    MicroPoint  fLetterSpace;       // letterspace for justification
    scXRect     fExtents;           // extents of chars
    TypeSpec    fRubiSpec;          // spec of rubi

            scRubiData(){}
            scRubiData( const UCS2 *, long, long, TypeSpec );

    void    TransOffsets( long );

    void    Read( APPCtxPtr, IOFuncPtr );
    void    PtrRestore( void );
    void    Write( APPCtxPtr, IOFuncPtr );
};


/* ================================================================ */

class scRubiArray : public scMemArray {
public:
            scRubiArray( );
            ~scRubiArray();

            // is there an annotation at this location
    Bool    IsRubiData( long );
    Bool    IsRubiData( long, long );

            // get the nth rubi data that occur between the
            // specified offset setting its position and the rubidata
    Bool    GetRubiAt( scRubiData&, long );
    Bool    GetNthRubi( int&, scRubiData&, int, long, long );

            // as we edit text change the annotations
    void    BumpRubiData( long, long );

            // place the rubiarray into the existing rubi array at the
            // indicated offset with it covering the number of chars
            // indicated
    void    Paste( const scRubiArray&, long offset, int size );

            // apply the style to the rubidata found within the bounds
```

```
/*==================================================================*/
```

09529370 .073400

```
                setmax = 0;
                break;

            case eNextLine:
                NextLine();
                setmax = 0;
                break;

            case eStartLine:
                StartLine();
                break;

            case eEndLine:
                EndLine();
                break;

            case ePrevPara:
            case eNextPara:
            case eFirstPara:
            case eLastPara:
                Para( moveSelect );
                break;

            case eBeginPara:
                BeginPara();
                break;
            case eEndPara:
                EndPara();
                break;

            case ePrevEntireColumn:
                PrevColumn();
                break;
            case eNextEntireColumn:
                NextColumn();
                break;

            case eBeginColumn:
                StartColumn( );
                break;
            case eEndColumn:
                EndColumn();
                break;

            case eStartStream:
                fMark.SelectStartStream();
                fPoint.SelectStartStream();
                break;

            case eEndStream:
                fMark.SelectEndStream();
                fPoint.SelectEndStream();
                break;

            default:
                SCDebugBreak();
                break;
        }
    fMark.UpdateInfo( setmax );
    fPoint.UpdateInfo( setmax );
}

/*======================================================================*/

void scSelection::NthPara( scStream* stream,
                           long      nthPara )
{
    scContUnit* p = stream->NthPara( nthPara );

    if ( p )
        SetParaSelection( p, 0, p->GetContentSize() );
}
```

```
        case eFirstPara:
            fPoint.SelectStartStream();
            break;

        case eLastPara:
            fPoint.SelectEndStream();
            break;

        case eStartStream:
            fPoint.SelectStartStream();
            break;

        case eEndStream:
            fPoint.SelectEndStream();
            break;

        case ePrevEntireColumn:
        case eNextEntireColumn:

        default:
            SCDebugBreak();
            break;
    }
    fMark.UpdateInfo( setmax );
    fPoint.UpdateInfo( setmax );
}

/*=================================================================*/

void scSelection::MoveSelect( eSelectMove moveSelect )
{
    int setmax = 1;

    switch ( moveSelect ) {
        case ePrevChar:
        case eNextChar:
        case ePrevCharInPara:
        case eNextCharInPara:
            SLCCharacterMove( *this, moveSelect );
            break;

        case ePrevWord:
            PrevWord( );
            break;
        case eNextWord:
            NextWord( );
            break;

        case ePrevSpellWord:
            PrevSpellWord( );
            break;
        case eNextSpellWord:
            NextSpellWord( );
            break;

        case eStartWord:
            StartWord( );
            break;

        case eEndWord:
            EndWord( );
            break;

        case ePrevEntireLine:
            PrevEntireLine();
            break;

        case eNextEntireLine:
            NextEntireLine();
            break;

        case ePrevLine:
            PrevLine();
```

```
                break;
            case eNextSpellWord:
                fPoint.SelectNextSpellWord( );
                break;

            case eStartWord:
                fPoint.SelectStartWord( );
                break;

            case eEndWord:
                fPoint.SelectEndWord( );
                break;

            case ePrevEntireLine:
                fPoint.SelectPrevLine();
                fPoint.SelectStartLine();
                break;

            case eNextEntireLine:
                fPoint.SelectNextLine();
                fPoint.SelectEndLine();
                break;

            case ePrevLine:
                fPoint.SelectPrevLine();
                setmax = 0;
                break;

            case eNextLine:
                fPoint.SelectNextLine();
                setmax = 0;
                break;

            case eStartLine:
                fPoint.SelectStartLine();
                break;

            case eEndLine:
                fPoint.SelectEndLine();
                break;

            case eBeginPara:
                fPoint.SelectStartPara();
                break;

            case eEndPara:
                fPoint.SelectEndPara();
                break;

            case eBeginColumn:
                fPoint.SelectStartColumn();
                break;

            case eEndColumn:
                fPoint.SelectEndColumn();
                break;

            case ePrevCharInPara:
                fPoint.SelectPrevCharInPara();
                break;

            case eNextCharInPara:
                fPoint.SelectNextCharInPara();
                break;

            case ePrevPara:
                fPoint.SelectPrevPara();
                break;

            case eNextPara:
                fPoint.SelectNextPara();
                break;
```

```
    sortedSelect.Sort();
    TextMarker& mark     = sortedSelect.fMark;
    TextMarker& point    = sortedSelect.fPoint;

    switch ( moveSelect ) {
        case ePrevChar:
            if ( !mark.fOffset && mark.fPara->GetPrev() ) {
                scContUnit* para = sortedSelect.fMark.fPara->GetPrev();
                select.SetParaSelection( para, PARAChSize( para ), PARAChSize( para ) );

                break;
            }
        case ePrevCharInPara:
            point.fOffset = MAX( 0, point.fOffset - 1 );
            mark = point;
            moved = sortedSelect.fMark != mark;
            select.SetMark( mark );
            select.SetPoint( point );
            break;

        case eNextChar:
            if ( point.fOffset == PARAChSize( point.fPara ) && point.fPara->GetNext() ) {
                scContUnit* para = sortedSelect.fMark.fPara->GetNext();
                select.SetParaSelection( para, 0, 0 );
                break;
            }
        case eNextCharInPara:
            point.fOffset = MIN( point.fOffset + 1, PARAChSize( point.fPara ) );
            mark = point;
            moved = sortedSelect.fPoint != point;
            select.SetMark( mark );
            select.SetPoint( point );
            break;
        default:
            break;
    }

    return moved;

/*==================================================================*/

void scSelection::Extend( eSelectMove moveSelect )
{
    int setmax = 1;

    switch ( moveSelect ) {
        case ePrevChar:
            fPoint.SelectPrevChar();
            break;
        case eNextChar:
            fPoint.SelectNextChar();
            break;

        case ePrevWord:
        {
            TextMarker mark = fMark;
            PrevWord();
            fMark = mark;
            break;
        }

        case eNextWord:
        {
            TextMarker mark = fMark;
            NextWord( );
            fMark = mark;
            break;
        }

        case ePrevSpellWord:
            fPoint.SelectPrevSpellWord( );
```

```
{
    scSelection sortedSelect( *this );

    sortedSelect.Sort();
    TextMarker& mark    = sortedSelect.fMark;
    TextMarker& point   = sortedSelect.fPoint;

    mark.fOffset = 0;
    point = mark;

    SetMark( mark );
    SetPoint( point );

    return mark.fPara != 0;
}

/*===================================================================*/

int scSelection::EndPara( )
{
    scSelection sortedSelect( *this );

    sortedSelect.Sort();
    TextMarker& mark    = sortedSelect.fMark;
    TextMarker& point   = sortedSelect.fPoint;

    point.fOffset = PARAChSize( point.fPara );
    mark = point;

    SetMark( mark );
    SetPoint( point );

    return mark.fPara != 0;
}

/*===================================================================*/

int scSelection::Para( eSelectMove  moveSelect )
{
    scSelection sortedSelection( *this );
    scContUnit* para;

    sortedSelection.Sort( );

    switch ( moveSelect ) {
        case ePrevPara:
            para = sortedSelection.fMark.fPara->GetPrev();
            break;
        case eNextPara:
            para = sortedSelection.fMark.fPara->GetNext();
            break;
        case eFirstPara:
            para = (scContUnit*)sortedSelection.fMark.fPara->FirstInChain();
            break;
        case eLastPara:
            para = (scContUnit*)sortedSelection.fMark.fPara->LastInChain();
            break;
    }

    if ( para )
        SetParaSelection( para, 0, PARAChSize( para ) );

    return para != 0;
}

/*===================================================================*/
// returns true if selection moved

static int SLCharacterMove( scSelection&   select,
                            eSelectMove    moveSelect )
{
    int moved = 0;
    scSelection sortedSelect( select );
    return mark.fCol != 0;
}

/*======================================================================*/

int scSelection::EndColumn(  )
{
    scSelection sortedSelect( *this );

    sortedSelect.Sort();
    TextMarker& mark    = sortedSelect.fMark;
    TextMarker& point   = sortedSelect.fPoint;

    if ( !point.fCol ) {
        scContUnit* lastPara    = point.fPara->GetPrevVisiblePara();
        scTextline* lastTxl     = 0;

        if ( lastPara )
            lastTxl = lastPara->GetLastVisibleLine();

        if ( lastPara && lastTxl ) {
            point.fCol  = lastTxl->GetColumn();
            point.fPara = lastPara;
            point.fTxl      = lastTxl;
        }
        else
            return 0;
    }

    point.SelectEndColumn( );
    mark = point;

    SetMark( mark );
    SetPoint( point );

    return mark.fCol != 0;
}

/*======================================================================*/

int scSelection::StartColumn( )
{
    scSelection sortedSelect( *this );

    sortedSelect.Sort();
    TextMarker& mark    = sortedSelect.fMark;
    TextMarker& point   = sortedSelect.fPoint;

    if ( !mark.fCol ) {
        scContUnit* lastPara    = mark.fPara->GetPrevVisiblePara();
        scTextline* txl         = lastPara->GetFirstline();
        if ( lastPara && txl ) {
            mark.fCol   = txl->GetColumn();
            mark.fPara  = lastPara;
            mark.fTxl   = txl;
        }
        else
            return 0;
    }

    mark.SelectStartColumn();

    point = mark;

    SetMark( mark );
    SetPoint( point );

    return mark.fCol != 0;
}

/*======================================================================*/

int scSelection::BeginPara(  )
```

```
        if ( lastPara && lastTxl )
            prevCol = lastTxl->GetColumn();
        else
            return 0;
    }
    else
        prevCol = mark.fCol->GetPrev();

        /* check to see if we have a next column */
    if ( !prevCol )
        return 0;

    if ( !prevCol->GetFirstline() )
        return 0;

    mark.fCol    = prevCol;
    mark.fTxl    = prevCol->GetFirstline();
    mark.fPara   = mark.fTxl->GetPara();

    point = mark;
    mark.SelectStartColumn();
    point.SelectEndColumn();

    SetMark( mark );
    SetPoint( point );

    return mark.fCol != 0;
}

/*====================================================================*/

int scSelection::NextColumn()
{
    scSelection sortedSelect( *this );

    sortedSelect.Sort();
    TextMarker& mark     = sortedSelect.fMark;
    TextMarker& point    = sortedSelect.fPoint;
    scColumn*   nextCol;

    if ( !point.fCol ) {
        scContUnit* lastPara     = point.fPara->GetPrevVisiblePara();
        scTextline* lastTxl      = lastPara->GetLastVisibleLine();
        if ( lastPara && lastTxl ) {
            point.fCol  = lastTxl->GetColumn();
            point.fPara = lastPara;
            point.fTxl      = lastTxl;
        }
        else
            return 0;
    }

    nextCol = point.fCol->GetNext();

        /* check to see if we have a next column */
    if ( !nextCol )
        return 0;

    if ( !nextCol->GetFirstline() )
        return 0;

    point.fCol = nextCol;
    point.fTxl = nextCol->GetFirstline();
    point.fPara = point.fTxl->GetPara();

    mark = point;
    mark.SelectStartColumn();
    point.SelectEndColumn();


    SetMark( mark );
    SetPoint( point );
```

```
        mark.fTxl = nextTxl;

        point = mark;
        mark.SelectStartLine();
        point.SelectEndLine();
    }

    SetMark( mark );
    SetPoint( point );

    return mark.fTxl != 0;
}

/*=================================================================*/

int scSelection::StartLine(  )
{
    scSelection sortedSelect( *this );

    sortedSelect.Sort();
    TextMarker& mark    = sortedSelect.fMark;
    TextMarker& point   = sortedSelect.fPoint;

    if ( mark.fTxl ) {
        mark.SelectStartLine();
        point = mark;
        SetMark( mark );
        SetPoint( point );
    }

    return mark.fTxl != 0;
}

/*=================================================================*/

int scSelection::EndLine( )
{
    scSelection sortedSelect( *this );

    sortedSelect.Sort();
    TextMarker& mark    = sortedSelect.fMark;
    TextMarker& point   = sortedSelect.fPoint;

    if ( point.fTxl ) {
        point.SelectEndLine( );
        mark = point;
    }
    else if ( mark.fTxl ) {
        mark.SelectEndLine();
        point = mark;
    }

    SetMark( mark );
    SetPoint( point );

    return mark.fTxl != 0;
}

/*=================================================================*/

int scSelection::PrevColumn()
{
    scSelection sortedSelect( *this );

    sortedSelect.Sort();
    TextMarker& mark    = sortedSelect.fMark;
    TextMarker& point   = sortedSelect.fPoint;
    scColumn*   prevCol;

    if ( !mark.fCol ) {
        scContUnit* lastPara    = mark.fPara->GetPrevVisiblePara();
        scTextline* lastTxl     = lastPara->GetLastVisibleLine();
        nextTxl = SearchRight( nextTxl, mark.fSelMaxX );

        if ( !nextTxl )
            return 0;

        point.fTxl = nextTxl;

        SelectLocateOnLine( &mark, eCursForward );
        point = mark;
    }

    SetMark( mark );
    SetPoint( point );
    return mark.fTxl != 0;
}

/*==================================================================*/

int scSelection::PrevEntireLine(  )
{
    scSelection sortedSelect( *this );

    sortedSelect.Sort();
    TextMarker& mark    = sortedSelect.fMark;
    TextMarker& point   = sortedSelect.fPoint;

    scTextline* prevTxl;

    if ( mark.fTxl ) {
        prevTxl = mark.fTxl->GetPrevLogical();
        if ( !prevTxl )
            return 0;

        mark.fTxl = prevTxl;

        point = mark;
        mark.SelectStartLine( );
        point.SelectEndLine( );
    }

    SetMark( mark );
    SetPoint( point );
    return mark.fTxl != 0;
}

/*==================================================================*/

int scSelection::NextEntireLine( )
{
    scSelection sortedSelect( *this );

    sortedSelect.Sort();
    TextMarker& mark    = sortedSelect.fMark;
    TextMarker& point   = sortedSelect.fPoint;

    scTextline* nextTxl;

    if ( point.fTxl ) {
        nextTxl = point.fTxl->GetNextLogical();
        if ( !nextTxl )
            return 0;

        point.fTxl = nextTxl;

        mark = point;
        mark.SelectStartLine();
        point.SelectEndLine();
    }
    else if ( mark.fTxl ) {
        nextTxl = mark.fTxl->GetNextLogical();
        if ( !nextTxl )
            return 0;
        if ( !prevTxl )
            return 0;

        mark.fTxl = prevTxl;

        SelectLocateOnLine( &mark, eCursBackward );
        point = mark;
    }
    else if ( point.fTxl ) {
        prevTxl = point.fTxl;
        do {
            prevTxl = prevTxl->GetPrevLogical();
        } while ( prevTxl && SameBaseline( point.fTxl, prevTxl ) );


        if ( !prevTxl )
            return 0;

        prevTxl = SearchLeft( prevTxl, point.fSelMaxX );

        if ( !prevTxl )
            return 0;
        point.fTxl = prevTxl;

        SelectLocateOnLine( &point, eCursBackward );
        mark = point;
    }

    SetMark( mark );
    SetPoint( point );
    return mark.fTxl != 0;
}

/*======================================================================*/

int scSelection::NextLine(  )
{
    scSelection sortedSelect( *this );

    sortedSelect.Sort();
    TextMarker& mark    = sortedSelect.fMark;
    TextMarker& point   = sortedSelect.fPoint;

    scTextline* nextTxl;

    if ( point.fTxl ) {
        nextTxl = point.fTxl;
        do {
            nextTxl = nextTxl->GetNextLogical();
        } while ( nextTxl && SameBaseline( point.fTxl, nextTxl ) );

        if ( !nextTxl )
            return 0;

        nextTxl = SearchRight( nextTxl, point.fSelMaxX );

        if ( !nextTxl )
            return 0;

        point.fTxl = nextTxl;

        SelectLocateOnLine( &point, eCursForward );
        mark = point;
    }
    else if ( mark.fTxl ) {
        nextTxl = mark.fTxl;
        do {
            nextTxl = nextTxl->GetNextLogical();
        } while ( nextTxl && SameBaseline( mark.fTxl, nextTxl ) );

        if ( !nextTxl )
            return 0;
```

```
static int SameBaseline( const scTextline* t1, const scTextline* t2 )
{
    return t1->GetBaseline() == t2->GetBaseline();
}

//#define SameBaseline( t1, t2 ) (t1)->GetOrigin().y == (t2)->GetOrigin().y

static scTextline* SearchLeft( scTextline* rightLineSegment,
                               MicroPoint selmax )
{
    scFlowDir fd = rightLineSegment->GetFlowdir();
    scMuPoint mPt;
    if ( fd.IsHorizontal() )
        mPt.Set( selmax, rightLineSegment->GetOrigin().y );
    else
        mPt.Set( rightLineSegment->GetOrigin().x, selmax );

    scTextline* prevTxl = 0;
    scXRect xrect;

    MicroPoint inflation = 0;
    for( int i = 0; i < 20; i++ ) {
        for( prevTxl = rightLineSegment;
             prevTxl && SameBaseline( prevTxl, rightLineSegment );
             prevTxl = prevTxl->GetPrev() ) {
                prevTxl->QueryExtents( xrect, 0 );
                xrect.Inset( inflation, 0 );
                if ( xrect.PinRect( mPt ) )
                    return prevTxl;
        }
        inflation -= scPOINTS( 4 );
    }
    return rightLineSegment;

/*==================================================================*/

static scTextline* SearchRight( scTextline* leftLineSegment,
                                MicroPoint  selmax )
{
    scTextline* nextTxl = 0;
    scTextline* rightLineSegment = 0;

    for( nextTxl = leftLineSegment;
         nextTxl && SameBaseline( nextTxl, leftLineSegment );
         nextTxl = nextTxl->GetNext() ) {
            rightLineSegment = nextTxl;
    }
    return SearchLeft( rightLineSegment, selmax );
}

/*==================================================================*/

int scSelection::PrevLine(  )
{
    scSelection sortedSelect( *this );

    sortedSelect.Sort();
    TextMarker& mark    = sortedSelect.fMark;
    TextMarker& point   = sortedSelect.fPoint;
    scTextline* prevTxl;

    if ( mark.fTxl ) {
        prevTxl = mark.fTxl;
        do {
            prevTxl = prevTxl->GetPrevLogical();
        } while ( prevTxl && SameBaseline( mark.fTxl, prevTxl ) );

        if ( !prevTxl )
            return 0;

        prevTxl = SearchLeft( prevTxl, mark.fSelMaxX );
    TextMarker& mark     = sortedSelect.fMark;
    TextMarker& point    = sortedSelect.fPoint;

    UCS2 ch = PARACharAtOffset( point.fPara, point.fOffset );
    if ( !CTIsSpace( ch ) )
        point.SelectEndSpellWord(  );

    if ( !IsSliverCursor() ) {
        while ( !point.SelectNextSpellWord(  ) ) {
            if ( scope == inContUnit )
                return 0;
            point.fPara = point.fPara->GetNext();
            if ( !point.fPara )
                return 0;
            point.fOffset = 0;
        }

        mark = point;

        if ( !mark.SelectStartSpellWord( ) || !point.SelectEndSpellWord( ) )
            return 0;
    }

    SetMark( mark );
    SetPoint( point );

    return mark.fPara != 0 && !IsSliverCursor();
}

/*==================================================================*/

int scSelection::StartWord( )

    scSelection sortedSelect( *this );

    sortedSelect.Sort();
    TextMarker& mark     = sortedSelect.fMark;
    TextMarker& point    = sortedSelect.fPoint;

    if ( !mark.SelectStartWord( ) )
        return 0;
    point = mark;

    SetMark( mark );
    SetPoint( point );
    return mark.fPara != 0;

/*==================================================================*/

int scSelection::EndWord(  )
{
    scSelection sortedSelect( *this );

    sortedSelect.Sort();
    TextMarker& mark     = sortedSelect.fMark;
    TextMarker& point    = sortedSelect.fPoint;

    if ( !point.SelectEndWord( ) )
        return 0;
    mark = point;

    SetMark( mark );
    SetPoint( point );

    return mark.fPara != 0;
}

/*==================================================================*/
/* LAYOUT BASED SELECTIONS - these need error checking
 * to see if layout exists
 */
/*==================================================================*/
```

```
    point.SelectStartWord( );

    SetMark( point );
    SetPoint( point );

    return point.fPara != 0;
}

/*==================================================================*/

int scSelection::NextWord( Scope scope )
{
    TextMarker point = fPoint;

    if ( CTIsSelectable( PARACharAtOffset( point.fPara, point.fOffset ) ) )
        point.SelectEndWord( );

    while ( !point.SelectNextWord( ) ) {
        if ( scope == inContUnit )
            return 0;
        point.fPara = point.fPara->GetNext();
        if ( !point.fPara )
            return 0;
        point.fOffset = 0;
    }
    point.SelectStartWord();

    SetMark( point );
    SetPoint( point );

    return point.fPara != 0;
}

/*==================================================================*/

int scSelection::PrevSpellWord( Scope scope )

    scSelection sortedSelect( *this );

    sortedSelect.Sort();
    TextMarker& mark    = sortedSelect.fMark;
    TextMarker& point   = sortedSelect.fPoint;

    if ( CTIsAlpha( PARACharAtOffset( mark.fPara, mark.fOffset ) ) )
        mark.SelectStartSpellWord( );

    while ( !mark.SelectPrevSpellWord( ) ) {
        if ( scope == inContUnit )
            return 0;
        mark.fPara = mark.fPara->GetPrev();
        if ( !mark.fPara )
            return 0;
        mark.fOffset = PARAChSize( mark.fPara );
    }

    point = mark;

    mark.SelectStartSpellWord( );
    point.SelectEndSpellWord( );

    SetMark( mark );
    SetPoint( point );

    return mark.fPara != 0;
}

/*==================================================================*/

int scSelection::NextSpellWord( Scope scope )
{
    scSelection sortedSelect( *this );

    sortedSelect.Sort();
```

```
int TextMarker::SelectStartColumn(  )
{
    scTextline* txl;

    scAssert( fCol != 0 );

    txl = fCol->GetFirstline();

    fTxl        = txl;
    fPara       = txl->GetPara();
    fParaCount  = fPara->GetCount();
    return SelectStartLine();
}

/*======================================================================*/

int TextMarker::SelectEndColumn(  )
{
    scTextline* txl;

    scAssert( fCol != 0 );

    txl = fCol->GetLastline( );

    /* tm->fCol should be correct */
    fPara       = txl->GetPara();
    fTxl        = txl;
    /* tm->colCount should be correct */
    fParaCount  = fPara->GetCount();
    fLineCount  = txl->GetLinecount();

    return SelectEndLine( );
}

/*======================================================================*/

int TextMarker::SelectStartStream()
{
    scStream* stream = fPara->GetStream();

    fPara = stream->First();
    fOffset = 0;
    return 1;
}

/*======================================================================*/

int TextMarker::SelectEndStream()
{
    scStream* stream = fPara->GetStream();

    fPara = stream->Last();
    fOffset = fPara->GetContentSize();
    return 1;
}


/*======================================================================*/
/* CONTENT BASED SELECTIONS */
/*======================================================================*/

int scSelection::PrevWord( Scope scope )
{
    TextMarker& point   = fPoint;

    while ( !point.SelectPrevWord() ) {
        if ( scope == inContUnit )
            return 0;
        point.fPara = point.fPara->GetPrev();
        if ( !point.fPara )
            return 0;
        point.fOffset = PARAChSize( point.fPara );
    }
    startChRec = (CharRecordP)fPara->GetCharArray().Lock();

    fOffset = TXTStartWord( startChRec, fOffset, eleminateLeadingSpaces );

    fPara->GetCharArray().Unlock();

    return fPara != 0;
}

/*=================================================================*/

int TextMarker::SelectEndSpellWord( )
{
    CharRecordP startChRec;

    startChRec = (CharRecordP)fPara->GetCharArray().Lock();

    fOffset = TXTEndWord( startChRec, fOffset );

    fPara->GetCharArray().Unlock();

    return fPara != 0;
}

/*=================================================================*/

int TextMarker::SelectStartPara()
{
    if ( fOffset == 0 )
        return 0;

    fOffset = 0;
    return 1;
}

/*=================================================================*/

int TextMarker::SelectEndPara()
{
    if ( fOffset == PARAChSize( fPara ) )
        return 0;

    fOffset = PARAChSize( fPara );
    return 1;
}

/*=================================================================*/

int TextMarker::SelectPrevPara()
{
    if ( !fPara->GetPrev() )
        return 0;

    fPara = fPara->GetPrev();
    SelectStartPara();
    return 1;
}

/* =============================================================== */

int TextMarker::SelectNextPara()
{
    if ( !fPara->GetNext() )
        return 0;

    fPara = fPara->GetNext();
    SelectEndPara();
    return 1;
}

/* =============================================================== */
```

```
    } while ( CTIsSpace( ch ) && endOffset > 0 );

    fOffset = endOffset;

    fPara->GetCharArray().Unlock();

    return !CTIsSpace( ch );
}

/*================================================================*/

int TextMarker::SelectNextSpellWord(  )
{
    CharRecordP startChRec;
    UCS2        ch;
    long        endOffset,
                limitOffset;

    limitOffset = PARAChSize( fPara );

    if ( fOffset >= limitOffset )
        return 0;

    startChRec = (CharRecordP)fPara->GetCharArray().Lock();

    endOffset   = fOffset;

    do {
        ch =  startChRec[endOffset++].character ;
    } while ( CTIsSpace( ch ) && endOffset <= limitOffset );

    fOffset = endOffset;

    fPara->GetCharArray().Unlock();

    return !CTIsSpace( ch ) && ch != scEndStream;
}

/*================================================================*/

int TextMarker::SelectStartWord(  )
{
    CharRecordP startChRec;

    startChRec = (CharRecordP)fPara->GetCharArray().Lock();

    fOffset = TXTStartSelectableWord( startChRec, fOffset );

    fPara->GetCharArray().Unlock();

    return fPara != 0;
}

/*================================================================*/

int TextMarker::SelectEndWord(  )
{
    scAssert( fOffset <= fPara->GetContentSize() );

    scHandleArrayLock   h( fPara->GetCharArray() );
    CharRecordP chRec = (CharRecordP)*h;

    fOffset = TXTEndSelectableWord( chRec, fOffset );

    return fPara != 0;
}

/*================================================================*/

int TextMarker::SelectStartSpellWord( int eleminateLeadingSpaces )
{
    CharRecordP startChRec;
    return fPara != 0 && fTxl != 0;
}

/*==================================================================*/

Bool TextMarker::SelectPrevWord(  )
{
    CharRecordP startChRec;
    UCS2        ch;
    long        endOffset;

    if ( !fOffset )
        return false;

    startChRec = (CharRecordP)fPara->GetCharArray().Lock();

    endOffset   = fOffset;

    do {
        ch =  startChRec[--endOffset].character ;
    } while ( !CTIsSelectable( ch ) && endOffset > 0 );

    fOffset = endOffset;

    fPara->GetCharArray().Unlock();
    return CTIsSelectable( ch );
}

/*==================================================================*/

Bool TextMarker::SelectNextWord(  )
{
    CharRecordP startChRec;
    UCS2        ch;
    long        endOffset,
                limitOffset;

    limitOffset = PARAChSize( fPara );

    if ( fOffset >= limitOffset )
        return false;

    startChRec = (CharRecordP)fPara->GetCharArray().Lock();

    endOffset   = fOffset;

    do {
        ch =  startChRec[endOffset++].character ;
    } while ( !CTIsSelectable( ch ) && endOffset <= limitOffset );

    fOffset = endOffset;

    fPara->GetCharArray().Unlock();
    return CTIsSelectable( ch );
}

/*==================================================================*/

Bool TextMarker::SelectPrevSpellWord( void )
{
    CharRecordP startChRec;
    UCS2        ch;
    long        endOffset;

    if ( !fOffset )
        return false;

    startChRec = (CharRecordP)fPara->GetCharArray().Lock();

    endOffset   = fOffset;

    do {
        ch =  startChRec[--endOffset].character ;
```

```
}

/*====================================================================*/

int TextMarker::SelectPrevLine()
{
    if ( !fTxl )
        return 0;

    if ( fTxl->GetPrevLogical() )
        fTxl = fTxl->GetPrevLogical();
    else
        return 0;

    SelectLocateOnLine( this, eCursBackward );
    return 1;
}

/*====================================================================*/

int TextMarker::SelectNextLine()
{
    if ( !fTxl )
        return 0;

    if ( fTxl->GetNextLogical() )
        fTxl = fTxl->GetNextLogical();
    else
        return 0;

    SelectLocateOnLine( this, eCursForward );
    return 1;
}

/* ================================================================== */

int TextMarker::SelectStartLine(  )
{
    scDebugAssert( fTxl != 0 );
    if ( fTxl ) {
        fCol        = fTxl->GetColumn();
        fPara       = fTxl->GetPara();

        fOffset     = fTxl->GetStartOffset( );
        fEndOfLine  = false;
    }

    return fPara != 0 && fTxl != 0;
}

/*====================================================================*/

int TextMarker::SelectEndLine(  )
{
    scDebugAssert( fTxl != 0 );
    if ( fTxl ) {
        fCol        = fTxl->GetColumn();
        fPara       = fTxl->GetPara();

#if 1
        if ( CTIsSpace( fTxl->CharAtEnd( ) ) ) {
            fOffset     = fTxl->GetEndOffset( ) - 1;
            fEndOfLine  = false;
        }
        else {
            fOffset     = fTxl->GetEndOffset( );
            fEndOfLine  = true;
        }
#else
        fOffset     = fTxl->GetEndOffset( );
        fEndOfLine  = true;
#endif
    }
```

```
/*================================================================*/

void scSelection::SetParaSelection( scContUnit* p,
                                    long        start,
                                    long        end )
{
    scStreamLocation    mark;
    scStreamLocation    point;
    scStream*           stream;

    stream          = p->GetStream();

    mark.fParaNum   = p->GetCount();
    AssertInRange( start, 0, p->GetContentSize() );
    mark.fParaOffset    = start;

    point = mark;
    AssertInRange( end, start, p->GetContentSize() );
    point.fParaOffset = end;

    Restore( &mark, &point, stream, true );
}

/*================================================================*/

int TextMarker::SelectPrevCharInPara()
{
    if ( !fOffset )
        return 0;
    fOffset -= 1;

    return 1;
}

/* ================================================================ */

int TextMarker::SelectNextCharInPara()
{
    if ( fOffset == fPara->GetContentSize() )
        return 0;

    fOffset = MIN( fPara->GetContentSize(), fOffset + 1 );
    return 1;
}

/* ================================================================ */

int TextMarker::SelectPrevChar()
{
    if ( !SelectPrevCharInPara() ) {
        if ( fPara->GetPrev() ) {
            fPara = fPara->GetPrev();
            fOffset = fPara->GetContentSize();
        }
        else
            return 0;
    }
    return 1;
}

/*================================================================*/

int TextMarker::SelectNextChar()
{
    if ( !SelectNextCharInPara() ) {
        if ( fPara->GetNext() ) {
            fOffset = 0;
            fPara = fPara->GetNext();
        }
        else
            return 0;
    }
    return 1;
/*==================================================================

    File:       scselec2.c

    $Header: /Projects/Toolbox/ct/Scselec2.cpp 4     6/18/97 10:20a Wmanis $

    Contains:   selection code

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

==================================================================*/


#include "scselect.h"
#include "scglobda.h"
#include "scparagr.h"
#include "scstream.h"
#include "sctextli.h"
#include "scctype.h"
#include "sccolumn.h"
#include "scmem.h"

#define AssertInRange( val, low, high )     scAssert( val >= low && val <= high );

static void SelectLocateOnLine( TextMarker *tm, eContentMovement movement )

    scMuPoint   mPt;
    Bool        vert        = false;
    scFlowDir   flowDir;

    tm->fPara   = tm->fTxl->GetPara();

    flowDir = tm->fTxl->GetFlowdir( );
    vert = flowDir.IsVertical();

    if ( vert ) {
        mPt.x   = tm->fTxl->GetOrigin().x;
        mPt.y   = tm->fSelMaxX;
    }
    else {
        mPt.x   = tm->fSelMaxX;
        mPt.y   = tm->fTxl->GetOrigin().y;
    }

    if ( vert )
        mPt.x -= tm->fTxl->GetVJOffset();
    else
        mPt.y += tm->fTxl->GetVJOffset();

    scXRect xrect;
    tm->fTxl->QueryExtents( xrect, 1 );

    xrect.Clamp( mPt );

    scMuPoint   charOrg;
    tm->fTxl->Select( charOrg, tm->fOffset, mPt, movement, tm->fEndOfLine );
    if ( vert )
        tm->fHLoc = charOrg.y;
    else
        tm->fHLoc = charOrg.x;
}
```

```
    else {
        endLocation = PARAChSize( startParaH );
        startParaH->Iter( func, startLocation, endLocation );

        for ( para = startParaH->GetNext(); para != endParaH; para = para->GetNext() ) {
            endLocation = PARAChSize( para );
            para->Iter( func, 0, endLocation );
        }

        endLocation     = fPoint.fOffset;
        endParaH->Iter( func, 0, endLocation );
        fPoint.fOffset = endLocation;
    }

    STRReformat( NULL, startParaH, scReformatTimeSlice, redispList );

    ValidateSelection( validatedSelection );
    UpdateSelection( );
}

/* ==================================================================== */

int32 scSelection::ContentSize() const
{
    if ( fMark.fPara == fPoint.fPara )
        return ABS( fMark.fOffset - fPoint.fOffset );
    else {
        // not implemented
        return LONG_MAX;
    }
}

/* ==================================================================== */

void scSelection::CopyAPPText( stTextImportExport& apptext )
{
    scScrapPtr scrap;

    CopyText( scrap );
    if ( scrap ) {
        ((scStream*)scrap)->CopyAPPText( apptext );

        long bytesFreed;
        ((scContUnit*)scrap)->FreeScrap( bytesFreed );
    }
}

/* ==================================================================== */

void scSelection::PasteAPPText( stTextImportExport& apptext,
                                scRedispList*       redisplist )
{
    TypeSpec ts;
    scStream* scrap = scStream::ReadAPPText( apptext );
    if ( scrap ) {
        PasteText( scrap, ts, redisplist );

        long bytesFreed;
        ((scContUnit*)scrap)->FreeScrap( bytesFreed );
    }
}

/* ==================================================================== */
```

```
    mark     = aRng.fMark.fPara;
    point    = aRng.fPoint.fPara;
}

/* ================================================================== */
/* create a selection point out of two ParaLocations */

void scSelection::Restore( const scStreamLocation*  mark,
                           const scStreamLocation*  point,
                           const scStream*          stream,
                           Bool                     geoChange )
{
    if ( mark ) {
        scStreamLocation tmark( *mark );
        SetLocationInfo( fMark, &tmark, (scStream*)stream, geoChange );
    }
    else
        fMark.Zero( theMARK, false );

    if ( point ) {
        scStreamLocation tpoint( *point );
        SetLocationInfo( fPoint, &tpoint, (scStream*)stream, geoChange );
    }
    else
        fPoint.Zero( thePOINT, false );
}

/* ================================================================== */
// return stream associate with selection

scStream* scSelection::GetStream( ) const
{
    return fMark.fPara->GetStream();
}

/* ================================================================== */

TypeSpec scSelection::GetSpecAtStart( void ) const
{
    scContUnit* firstContUnit;

    firstContUnit = fMark.fPara->Earlier( fPoint.fPara );
    if ( firstContUnit == fMark.fPara )
        return fMark.fPara->SpecAtOffset( fMark.fOffset );

    return fPoint.fPara->SpecAtOffset( fPoint.fOffset );
}

/* ================================================================== */

void    scSelection::Iter( SubstituteFunc   func,
                           scRedispList*    redispList )
{
    scSelection     validatedSelection;
    scContUnit  *para,
                *startParaH,
                *endParaH;
    long        startLocation,
                endLocation;

    raise_if( !ValidateSelection( validatedSelection ), scERRlogical );

    Sort( );
    startParaH      = fMark.fPara;
    endParaH        = fPoint.fPara;
    startLocation   = fMark.fOffset;

    if ( startParaH == endParaH ) {
        endLocation     = fPoint.fOffset;
        startParaH->Iter( func, startLocation, endLocation );
        fPoint.fOffset = endLocation;
    }
    pl.fParaOffset  = tm.fOffset;
    pl.fEndOfLine   = tm.fEndOfLine;

    pl.fParaSpec    = tm.fPara->GetDefaultSpec();
    tm.fPara->ChInfo( tm.fOffset, pl.fTheCh, pl.fFlags,
                      pl.fTheChWidth, pl.fWordSpace,
                      pl.fChSpec, pl.fUnitType );

    if ( tm.fPara->FindLocation( tm.fOffset, tm.fEndOfLine, txl, fHLoc, eCursNoMovement ) )
        pl.fAPPColumn = txl->GetColumn()->GetAPPName( );
    else {
        if ( pl.fStream->FindColumn( col ) )
            pl.fAPPColumn = ((scColumn*)col->LastInChain())->GetAPPName();
        else
            pl.fAPPColumn = 0;
    }
    pl.fPosOnLine       = ( txl ? tm.fHLoc : LONG_MIN );
    pl.fSelMaxX         = ( txl ? tm.fSelMaxX : LONG_MIN );

    if ( pl.fChSpec.ptr() ) {
        pl.fFont            = scCachedStyle::GetCachedStyle( pl.fChSpec ).GetFont();
        pl.fPointSize       = scCachedStyle::GetCachedStyle( pl.fChSpec ).GetPtSize();
    }
    else {
        SCmemset( &pl.fFont, 0, sizeof( APPFont ) );
        pl.fPointSize       = 0;
    }

    if ( txl == 0 ) {
        pl.fBaseline    = LONG_MIN;
    }
    else {
        pl.fBaseline    = txl->GetBaseline();
        pl.fBaseline    += txl->GetVJOffset();
    }

    pl.fMeasure     = ( txl ? txl->GetLength() : LONG_MIN );
    pl.fLetterSpace = ( txl ? txl->GetLSP() : LONG_MIN );
}

/* ================================================================== */
/* decompose the selection into two seperate structures that
 * contain information about the selection
 */

void scSelection::Decompose( scStreamLocation&  mark,
                             scStreamLocation&  point )
{
    scSelection aRng( *this );
    aRng.Sort( );       /* sort the selection */

    GetLocationInfo( aRng.fMark, mark );
    GetLocationInfo( aRng.fPoint, point );
}

/* ================================================================== */

void scSelection::Decompose2( scStreamLocation& mark,
                              scStreamLocation& point )
{
    scSelection aRng( *this );

    GetLocationInfo( aRng.fMark, mark );
    GetLocationInfo( aRng.fPoint, point );
}

/* ================================================================== */

void scSelection::GetContUnits( scContUnit*& mark,
                                scContUnit*& point ) const
{
    scSelection aRng( *this );
    aRng.Sort( );       /* sort the selection */
```

```
        fPoint.fPara = fPoint.fPara->PasteParas( scrapCopy, fPoint.fOffset );
    else
        fMark.fPara->PasteText( scrapCopy, fPoint.fOffset );

    ((scStream*)scrapCopy)->STRFree();

    SLCRecomposeLogical( firstCol, fMark.fPara, redisplist, this, true );
    ChangeSelection( SLC_LOCATE, false );
}

/* ==================================================================== */
/* create part of a selection by using the information from a paralocation
 * probably being called from SLCRecompose
 */

static void SetLocationInfo( TextMarker&        tm,
                             scStreamLocation*  pl,
                             scStream*          stream,
                             Bool               geoChange )
{
    scTextline* txl;
    MicroPoint  fHLoc;

    if ( stream )
        pl->fStream = stream;

    tm.fPara          = pl->fStream->NthPara( pl->fParaNum );
    if ( tm.fPara ) {
        tm.fParaCount    = pl->fParaNum;
        tm.fOffset       = pl->fParaOffset;
        tm.fEndOfLine    = pl->fEndOfLine;
    }
    else {
        tm.fPara = pl->fStream->Last( );
        tm.fParaCount    = tm.fPara->GetCount();
        tm.fOffset       = 0;
        tm.fEndOfLine    = false;
    }

    if ( tm.fPara->FindLocation( tm.fOffset, tm.fEndOfLine, txl, fHLoc, eCursNoMovement ) ) {
        tm.fCol          = txl->GetColumn();
        tm.fTxl          = txl;
        tm.fColCount     = tm.fCol->GetCount( );
        tm.fLineCount    = txl->GetLinecount();
        tm.fHLoc         = fHLoc;
        if ( geoChange )
            tm.fSelMaxX      = tm.fHLoc;
        else
            tm.fSelMaxX      = pl->fSelMaxX;
    }
    else {
        tm.fCol          = 0;
        tm.fTxl          = 0;
        tm.fColCount     = -1;
        tm.fLineCount    = -1;
        tm.fHLoc         = LONG_MIN;
        tm.fSelMaxX      = LONG_MIN;
//      tm.fFlowDir.Invalidate();
    }
}

/* ==================================================================== */
// get information about this location within the text

static void GetLocationInfo( TextMarker& tm, scStreamLocation& pl )
{
    scTextline* txl;
    MicroPoint  fHLoc;
    scColumn*       col;

    pl.fStream       = tm.fPara->GetStream();
    pl.fParaNum      = tm.fPara->GetCount();
```

```
                *para2,
                *lastPara;
    long        offset1,
                offset2;

    try {
        scSelection aRng( *this );
        aRng.Sort( );

        if ( aRng.fMark.fPara &&  aRng.fPoint.fPara ) {
            if ( aRng.fMark.fPara == aRng.fPoint.fPara )
                scrap = aRng.fMark.fPara->CopyText( aRng.fMark.fOffset, aRng.fPoint.fOffset );
            else {
                para1   = aRng.fMark.fPara;
                offset1 = aRng.fMark.fOffset;
                para2   = aRng.fPoint.fPara;
                offset2 = aRng.fPoint.fOffset;

                lastPara = para2->GetNext( );
                for ( para = para1; para && para != lastPara; para = para->GetNext() ) {
                    if ( para == para1 )
                        scrap = para->CopyText( offset1, LONG_MAX );
                    else if ( para == para2 )
                        ((scContUnit*)scrap)->Append( para->CopyText( LONG_MIN, offset2 ) );
                    else
                        ((scContUnit*)scrap)->Append( para->CopyText( LONG_MIN, LONG_MAX ) );
                }
            }
        }
    }
    catch( ... ) {
        ((scStream*)scrap)->STRFree(), scrap = 0;
        throw;
    }
}

/* ================================================================ */
/* paste text into the selection applying the indicated style */

void scSelection::PasteText( const scStream*    scrap,
                             TypeSpec           style,
                             scRedispList*      redisplist )
{
    scColumn*   firstCol    = fMark.fCol;
    scContUnit* para;
    scStream*   scrapCopy;

    if ( fMark.fPara->GetFirstline() )
        firstCol = fMark.fPara->GetFirstCol();
    else
        firstCol = fMark.fCol;

    if ( fMark != fPoint )
        ClearText( redisplist, false );
    scAssert( fMark == fPoint );

    /* the pasted text may have no spec associated with it, we need to
     * associate a spec with it and to retabluate if necessary
     */

    scrap->STRCopy( scrapCopy );

    if ( style.ptr() ) {
        for ( para = scrapCopy; para; para = para->GetNext( ) )
            para->SetStyle( 0, LONG_MAX, style, true, false );
    }

    TypeSpec nullSpec;

    for ( para = scrapCopy; para; para = para->GetNext( ) )
        para->Retabulate( nullSpec );

    if ( scrapCopy->GetNext() )
    aRng.Sort( );

    if ( aRng.fMark.fPara->GetFirstline() )
        firstCol = aRng.fMark.fPara->GetFirstCol();
    else
        firstCol = aRng.fMark.fCol;

    if ( aRng.fMark.fPara == aRng.fPoint.fPara ) {
        firstPara = aRng.fMark.fPara;
         aRng.fMark.fPara->ClearText( aRng.fMark.fOffset, aRng.fPoint.fOffset );
    }
    else {
        para1   = aRng.fMark.fPara;
        offset1 = aRng.fMark.fOffset;
        para2   = aRng.fPoint.fPara;
        offset2 = aRng.fPoint.fOffset;

        /* if there will be text left over, then we will
         * merge the paragraphs
         */
        merge = ( offset1 && offset2 < PARAChSize( para2 ) );

        lastPara = para2->GetNext( );
        for ( para = para1; para && para != lastPara; para = nextPara ) {
            nextPara = para->GetNext( );
            if ( para == para1 ) {
                if ( offset1 == 0 ) {
                    aRng.CorrectSelection( para1, offset1, para2, offset2 );
                    merge = true;
                }
                para->ClearText( offset1, LONG_MAX );
            }
            else if ( para == para2 ) {
                if ( para->ClearText( LONG_MIN, offset2 ) ) {
                    para->Unlink( );
                    para->Free( );
                    merge = false;
                }
            }
            else {
                para->ClearText( LONG_MIN, LONG_MAX );
                para->Unlink( );
                para->Free( );
            }
        }

        firstPara = para1;
        para1->Renumber();

        /* after this the paragraphs in the selection may be invalid */

        if ( merge ) {
            offset1 = 0;
            para2 = para2->Merge( offset1 );
        }
    }

    aRng.fPoint = aRng.fMark;
    *this = aRng;

    if ( repair ) {
        SLCRecomposeLogical( firstCol, firstPara, redispList, &aRng, true );
        ChangeSelection( SLC_LOCATE, false );
    }
}

/* ================================================================== */
/* copy the text contained with the selection */

void scSelection::CopyText( scScrapPtr& scrap )
{
    scContUnit  *para,
                *para1,
    scContUnit* para1H;
    scContUnit* para2H;

    aRng.Sort( );
    col1H       = aRng.fMark.fCol;

    para1H      = aRng.fMark.fPara;
    offset1     = aRng.fMark.fOffset;
    para2H      = aRng.fPoint.fPara;
    offset2     = aRng.fPoint.fOffset;

    scAssert( para1H == para2H );

    para1H->ApplyAnnotation( offset1, offset2, annotation );

    STRReformat( col1H, para1H, scReformatTimeSlice, redispList );
    ChangeSelection( SLC_LOCATE, false );
}

#endif

/* ==================================================================== */
/* correct the selection, something probably has changed and we need to
 * correct the state contained in the selection
 */

void scSelection::CorrectSelection( scContUnit* para1,
                                    long        ,
                                    scContUnit* para2,
                                    long        offset2 )
{
    /* we will end up blasting this paragraph so lets take
     * care of the selection range
     */
    if ( offset2 < PARAChSize( para2 ) ) {
        fMark.fPara         = para1;
        fMark.fOffset       = 0;
        fMark.fEndOfLine    = false;
    }

    fPoint = fMark;
}

/* ==================================================================== */
/* cut the text contained within the selection */

/* cut the text contained within the selection */

void scSelection::CutText( scScrapPtr&      scrap,
                           scRedispList*    redispList )
{
    CopyText( scrap );
    ClearText( redispList, true );
}

/* ==================================================================== */
/* clear any text contained within the selection */

void scSelection::ClearText( scRedispList*  redispList,
                             Bool           repair )
{
    scColumn*   firstCol;
    scContUnit* firstPara;
    scContUnit* para;
    scContUnit* para1;
    scContUnit* para2;
    scContUnit* nextPara;
    scContUnit* lastPara;
    long        offset1,
                offset2;
    Bool        merge;

    scSelection aRng( *this );
    para1H      = aRng.fMark.fPara;
    offset1     = aRng.fMark.fOffset;
    para2H      = aRng.fPoint.fPara;
    offset2     = aRng.fPoint.fOffset;

    /* walk thru the paras transforming them */
    if ( para1H && para2H ) {
        if ( para1H == para2H )
            para1H->TextTrans( offset1, offset2, trans, numChars );
        else {
            lastPara = para2H->GetNext( );

            for ( para = para1H;
                        para && para != lastPara;
                        para = para->GetNext() ) {

                if ( para == para1H )
                    para->TextTrans( offset1, LONG_MAX, trans, numChars );
                else if ( para == para2H )
                    para->TextTrans( LONG_MIN, offset2, trans, numChars );
                else
                    para->TextTrans( LONG_MIN, LONG_MAX, trans, numChars );
            }
        }
    }

    /* force repaint of range */
    aRng.MarkValidatedSelection( scREPAINT );

    /* reformat */
    if ( para1H )
        STRReformat( col1H, para1H, scReformatTimeSlice, redispList );

    /* correct selection */
    ChangeSelection( SLC_LOCATE, false );

/* ==================================================================== */
#ifdef _RUBI_SUPPORT

Bool scSelection::GetAnnotation( int            nth,
                                 scAnnotation&  annotation )
{
    scSelection aRng( *this );
    scColumn        *col1H;
    long        offset1,
                offset2;
    scContUnit  *p1,
                *p2;

    aRng.Sort( );
    col1H       = aRng.fMark.fCol;

    p1          = aRng.fMark.fPara;
    offset1     = aRng.fMark.fOffset;
    p2          = aRng.fPoint.fPara;
    offset2     = aRng.fPoint.fOffset;

    scAssert( p1 == p2 );

    return p1->GetAnnotation( nth, offset1, offset2, annotation );
}

/* ==================================================================== */

void scSelection::ApplyAnnotation( const scAnnotation&  annotation,
                                   scRedispList*        redispList )
{
    scSelection aRng( *this );
    scColumn*   col1H;
    long        offset1,
                offset2;
```

```
{
    scColumn*   col1;
    long        offset1,
                offset2;
    scContUnit* para;
    scContUnit* para1;
    scContUnit* para2;
    scContUnit* lastPara;

    scSelection aRng( *this );
    aRng.Sort( );

    if ( aRng.fMark.fPara->GetFirstline() )
        col1 = aRng.fMark.fPara->GetFirstCol();
    else
        col1 = aRng.fMark.fCol;

    para1   = aRng.fMark.fPara;
    offset1 = aRng.fMark.fOffset;
    para2   = aRng.fPoint.fPara;
    offset2 = aRng.fPoint.fOffset;

        // force repaint of range
    aRng.MarkValidatedSelection( scREPAINT );

    if ( para1 && para2 ) {
        if ( para1 == para2 )
            para1->SetStyle( offset1, offset2, style, true, false );
        else {
            lastPara = para2->GetNext( );
            for (para = para1; para && para != lastPara; para = para->GetNext( ) ) {
                if (para == para1 )
                    para->SetStyle( offset1, LONG_MAX, style, true, false );
                else if ( para == para2 )
                    para->SetStyle( 0, offset2, style, true, false );
                else
                    para->SetStyle( 0, LONG_MAX, style, true, false );
            }
        }
    }

    if ( para1 )
        STRReformat( col1, para1, scReformatTimeSlice, redispList );

    ChangeSelection( SLC_LOCATE, false );
}
/* ==================================================================== */
/* apply the character transformation to the selection and return the
 * damage
 */

void scSelection::TextTrans( eChTranType    trans,
                             int            numChars,
                             scRedispList*  redispList )
{
    scSelection aRng( *this );
    scColumn        *col1H;
    long        offset1,
                offset2;
    scContUnit  *para,
                *para1H,
                *para2H,
                *lastPara;

        // set my world up
    aRng.Sort( );

    if ( aRng.fMark.fPara->GetFirstline() )
        col1H = aRng.fMark.fPara->GetFirstCol();
    else
        col1H = aRng.fMark.fCol;
```

```
            // SLCChangeSelection will reset fEndOfLine to false when it calls
            // SLCSetTextMarker, which calls ParaFindLocation.
            //
         aRng.fMark.fEndOfLine = aRng.fPoint.fEndOfLine = savedEndOfLine;
         aRng.ChangeSelection( SLC_LOCATE, false );
    }

         // set up data for immediate redisplay
    immediateRedisp.fStopLine = COLLineNum( &aRng );
    if ( firstColCursor != aRng.fMark.fCol )
        immediateRedisp.fStartLine = immediateRedisp.fStopLine;

    if ( immediateRedisp.fStopLine < immediateRedisp.fStartLine ) {
        short tmp                  = immediateRedisp.fStartLine;
        immediateRedisp.fStartLine = immediateRedisp.fStopLine;
        immediateRedisp.fStopLine  = tmp;
    }

    if ( aRng.fMark.fCol && rebreak ) {
        aRng.fMark.fCol->LineExtents( immediateRedisp );
        if ( redispList )
            redispList->SetImmediateRect( aRng.fMark.fCol, immediateRedisp );
    }

    if ( firstColCursor && firstColCursor->MoreText( ) ) {
                //
                // this is to insure that if the addition of a character
                // overflowed the box, but caused no damage we would
                // report back that
                // there was more text in the column
                //
            //stat = COLColRectAdd( firstColCursor, "SLCKeyArray2" );
    }
    else if ( !firstColCursor && aRng.fMark.fCol ) {
                //
                // this insures that if we transition from being in no
                // container to being in a container and there is no
                // damage we will report accurately more text
                //
            //stat = COLColRectAdd( aRng.fMark.fCol, "SLCKeyArray2" );
    }

    gStreamChangeInfo.Set( 0, 0, 0, 0 );
    *this = aRng;
}

/* ================================================================ */

void scSelection::InsertField( const clField&   field,
                               TypeSpec&        spec,
                               scRedispList*    redisp )
{
    CharRecord ch;
    ch.character    = scField;
    ch.flags.SetField( field.id() );

    scContUnit* para = fMark.fPara;
    para->Insert( ch, spec, fMark.fOffset );

    scColumn*   col;
    if ( para->GetFirstline() )
        col = para->GetFirstCol();
    else
        col = fMark.fCol;

    STRReformat( col, para, scReformatTimeSlice, redisp );
    MoveSelect( eNextChar );
}

/* ================================================================ */

void scSelection::SetStyle( TypeSpec        style,
                            scRedispList*   redispList )
                if ( rebreak ) {
                        // Restore end of line setting after adding
                        // characters in case we are still at the end
                        // of a hyphenated line. If we aren't,
                        // SLCChangeSelection will reset fEndOfLine to false
                        // when it calls SLCSetTextMarker, which calls
                        // ParaFindLocation.
                        //
                    aRng.fMark.fEndOfLine = aRng.fPoint.fEndOfLine = savedEndOfLine;
                    aRng.ChangeSelection( SLC_LOCATE, false );
                    savedEndOfLine  = aRng.fMark.fEndOfLine;
                }
                aRng.Decompose( keyRecords[count].mark(), dummyPoint );
                keyRecords[count].restoreselect() = true;
            }
            rebreak = false;
        }           /* at end of hyphenated line                */

        if ( para == firstPara && aRng.fMark.fOffset == 0 && keyRecords[count].keycode() == scBackSp
ace ) {
                // we are going to merge with previous para
            firstPara = 0;
        }
            // if key is backspace at start of stream, this will
            // set noOp field of key record
            //
        TextMarker savedPosition = aRng.fMark;

        para = aRng.fMark.fPara;
        Bool iAmRemoved = false;

        aRng.fMark.fPara = para->KeyInsert( aRng.fMark.fOffset,
                                             keyRecords[count],
                                             tmMove,
                                             rebreak,
                                             textCleared,
                                             clearedSpec,
                                             iAmRemoved );
        if ( iAmRemoved )
            para = 0;

        firstPara = aRng.fMark.fPara->Earlier( firstPara ? firstPara : para );
        aRng.fPoint = aRng.fMark;


        if ( !keyRecords[count].restoreselect() ) {
            if ( CMcontent( keyRecords[count].keycode() ) ) {
                selectionMoved  = aRng.ChangeSelection( tmMove, false );
            }
            else {
                selectionMoved  = aRng.ChangeSelection( tmMove, true );
                if ( !selectionMoved )
                    keyRecords[count].noop() = true;
            }
        }
        else {
            aRng.fPoint = aRng.fMark = savedPosition;
        }
    }

    aRng.fPoint = aRng.fMark;

    eRefEvent refevent = SLCRecomposeLogical( firstColPara, firstPara, redispList, &aRng, rebreak );
//  if ( refevent == eNoReformat )
//      throw( scNoReformat );

//  if ( rebreak && firstPara )
//      firstPara->scAssertValid();

    if ( rebreak ) {
            //
            // Restore end of line setting after adding characters in case
            // we are still at the end of a hyphenated line. If we aren't,
        gStreamChangeInfo.Set( 0, 0, 0, 0 );
    }

        //
        // if the selection is multiple chars clear it, it will be the
        // app's responsibility to clear this ahead of time for undoing
        //
    if ( aRng.fMark.fPara != aRng.fPoint.fPara || aRng.fMark.fOffset != aRng.fPoint.fOffset ) {
        if ( CMcontent( keyRecords[0].keycode() ) ) {

            #if SCDEBUG > 1
                {
                    static int doit = 0;
                    if ( doit ) {
                        aRng.fMark.fPara->GetSpecRun().PrintRun( "scSelection::KeyArray" );
                        SCDebugTrace( 0, "offsets %d %d\n",
                                      aRng.fMark.fOffset, aRng.fPoint.fOffset );
                    }
                }
            #endif

            clearedSpec = aRng.fMark.fPara->SpecAtOffset( aRng.fMark.fOffset + 1 );
            aRng.ClearText( redispList, true );
            textCleared = true;

            #if SCDEBUG > 1
                {
                    static int doit;
                    if ( doit )
                        aRng.fMark.fPara->GetSpecRun().PrintRun( "scSelection::KeyArray 2" );
                }
            #endif

        }
        else {
            switch ( keyRecords[0].keycode() ) {
                case scLeftArrow:
                case scUpArrow:
                    break;
                case scRightArrow:
                case scDownArrow:
                    aRng.fMark = aRng.fPoint;
                    break;
            }
        }
    }

    firstPara        = 0;

        // Remember this in case we add characters
        // right at the hyphen point.
    savedEndOfLine  = aRng.fMark.fEndOfLine;

        // walk down the array clearing text
    for ( count = 0; keyCount--; count++ ) {

        if ( keyRecords[count].noop() )
            continue;

        if ( ! CMcontent( keyRecords[count].keycode() ) ) {

            if (rebreak ) {
                SLCRecomposeLogical( firstColPara, firstPara, redispList, &aRng, rebreak );
                firstPara = 0;
            }

            if ( keyRecords[count].restoreselect() ) {
                Restore( &keyRecords[count].mark(), &keyRecords[count].mark(), 0, false );
                aRng = *this;
                aRng.Sort( );
                continue;
            }
            else {
```

```
            if ( aRng->fPoint.fPara ) {
                txl = aRng->fPoint.fPara->GetFirstline();
                if ( txl == 0 ) {
                    if ( ( prevPara = aRng->fPoint.fPara->GetPrev() ) != 0)
                        txl = prevPara->GetFirstline();
                    if ( !txl )
                        return eNoReformat;
                }
                firstCol = txl->GetColumn();
            }
            else
                return eNoReformat; /* !! */
        }
    }

    if ( rebreak && firstPara )
        return STRReformat( firstCol, firstPara, scInteractiveTimeSlice, redispList );
    return eNoReformat;
}

/* ==================================================================== */

static int GetOffsetChange( short           keyCount,
                            scKeyRecord*    keyRecords )
{
    int offset = 0;

    for ( ; keyCount--; )
        offset += CMcontent( keyRecords[keyCount].keycode() );
    return offset;
}

/* ==================================================================== */

void scSelection::KeyArray( short           keyCount,
                            scKeyRecord*    keyRecords,
                            scRedispList*   redispList )
{
    scSelection         aRng( *this );
    scColumn*           firstColPara;
    scColumn*           firstColCursor;
    scContUnit*         para            = 0;
    scContUnit*         firstPara;          /* para to start reformating at */
    scStreamLocation    dummyPoint;
    long                tmMove;
    register            count;
    short               rebreak         = false;
    Bool                textCleared     = false;
    Bool                selectionMoved;
    Bool                savedEndOfLine;
    scImmediateRedisp   immediateRedisp;
    TypeSpec            clearedSpec;

    aRng.Sort( );

    if ( aRng.fMark.fPara->GetFirstline( ) )
        firstColPara = aRng.fMark.fPara->GetFirstCol();
    else
        firstColPara = aRng.fMark.fCol;

    firstColCursor = aRng.fMark.fCol;

        // store what line we started on
    if ( firstColCursor ) {
        immediateRedisp.fStartLine  = COLLineNum( &aRng );
        immediateRedisp.fStopLine   = immediateRedisp.fStartLine;
        immediateRedisp.fImmediateRect.Set( 0, 0, 0, 0 );
        firstColCursor->LineExtents( immediateRedisp );
        gStreamChangeInfo.Set( firstColCursor, aRng.fMark.fPara, aRng.fPoint.fOffset, GetOffsetChang
e( keyCount, keyRecords ) );
    }
    else {
        immediateRedisp.fStartLine = -1;
        switch ( cSrtRng.RangeMovement( ) ) {
            case eBeforeSelect:
                    /* Extending line(s) from beginning of selection    */
            case eEqualSelect:
                    /* Neither adding nor subtracting lines.            */
                cSrtRng.LineHilite( func );
                break;
            case eAfterSelect:
                    /* Subtracting line(s) from beginning of selection  */
                cSrtRng.fPoint.fHLoc    = LONG_MAX;
                cSrtRng.LineHilite( func );
                APPDrawContext( fPoint.fCol->GetAPPName( ), fPoint.fCol, drawCtx );
                fPoint.fTxl->Hilite( NULL, LONG_MAX,
                                     &fPoint, fPoint.fHLoc,
                                     drawCtx,
                                     func, cSrtRng );
                break;
        }
    }
}

/* ==================================================================== */
/* Determine how this interactive selection is moving.  */
/* This is only called by SLCInteractiveHilite.         */
/* THIS ASSUMES MARK AND POINT ARE ON DIFFERENT LINES.  */

eSelectMovement scSelection::RangeMovement( ) const
{
    if ( fMark.fParaCount > fPoint.fParaCount )
        return eBeforeSelect;
    else if ( fMark.fParaCount == fPoint.fParaCount ) {
        if ( fMark.fOffset > fPoint.fOffset )
            return eBeforeSelect;
        else if ( fMark.fOffset == fPoint.fOffset )
            return eEqualSelect;
    }

    return eAfterSelect;
}

/* ==================================================================== */

static eRefEvent SLCRecomposeLogical( scColumn*     firstCol,
                                      scContUnit*   firstPara,
                                      scRedispList* redispList,
                                      scSelection*  aRng,
                                      short         rebreak )
{
    scTextline* txl;
    scContUnit* prevPara;

    /* the first column may be NULL if the selection were outside the
     * visible world, the SLCChangeSelection should may have put us back into
     * the visible world so let's check, otherwise no reformatting will
     * take place, there may be conditions in this type of situtation
     * that may create anomalies, I don't know what they are as of yet
     */

    if ( firstCol == 0 ) {
        /* there is a likelyhood here that we will
         * have to do nothing, since we may be out
         * of visible range, but then again this may
         * be forcing us back in so this may be tricky
         *
         * if the current paragraph doesn't have a line
         * lets check the previous paragraph to see if it
         * has a line - if it does we can start with that
         * column
         */

        firstCol = aRng->fPoint.fCol;
        if ( firstCol == 0 ) {
            // turn off old hiliting
        APPDrawContext( pSrtRng.fPoint.fCol->GetAPPName(), pSrtRng.fPoint.fCol, drawCtx );
        pSrtRng.fPoint.fTxl->Hilite( &pSrtRng.fPoint, pSrtRng.fPoint.fHLoc,
                                     &pSrtRng.fMark, pSrtRng.fMark.fHLoc,
                                     drawCtx, func, pSrtRng );

            /* turn on new hiliting  */
        LineHilite( func );
    }
    else if ( pSrtRng.fMark.fTxl==cSrtRng.fPoint.fTxl
                        || pSrtRng.fPoint.fTxl==cSrtRng.fMark.fTxl ) {
        /* prev selection was behind mark and new selection is past mark */

            /* turn off old hiliting */
        prevRange.LineHilite( func );

            /* turn on new hiliting  */
        LineHilite( func );
    }
    else if ( prevRange.fPoint.fTxl == pSrtRng.fPoint.fTxl ) {
            /* Prev selection more than one line, and we haven't flipped it.
             * Thus, the point succeeds the mark in both the previous and
             * current selections.
             */

            /* dehilite the line we were on */
        APPDrawContext( pSrtRng.fPoint.fCol->GetAPPName(), pSrtRng.fPoint.fCol, drawCtx );
        pSrtRng.fPoint.fTxl->Hilite( NULL, LONG_MIN, &pSrtRng.fPoint, pSrtRng.fPoint.fHLoc, drawCtx,
func, pSrtRng );

            /* we want to hilite from the old point to the new point,
             * including the line containing the old point
             */
        cSrtRng.fMark       = prevRange.fPoint;
        cSrtRng.fMark.fHLoc = LONG_MIN;
        cSrtRng.fPoint      = fPoint;

        switch ( cSrtRng.RangeMovement( ) ) {
            case eBeforeSelect: /* Backing up, dehiliting bottom line(s)    */
                cSrtRng.fPoint.fHLoc    = LONG_MIN;
                cSrtRng.LineHilite( func );

                APPDrawContext( fPoint.fCol->GetAPPName( ), fPoint.fCol, drawCtx );
                fPoint.fTxl->Hilite( NULL, LONG_MIN,
                                     &fPoint, fPoint.fHLoc,
                                     drawCtx, func, cSrtRng );
                break;
            case eEqualSelect:  /* Neither adding nor subtracting lines.    */
            case eAfterSelect:  /* Adding line(s) to bottom of selection    */
                cSrtRng.LineHilite( func );
                break;
        }
    }
    else if ( prevRange.fPoint.fTxl == pSrtRng.fMark.fTxl ) {
            /* Previous selection more than one line, and we haven't
             * flipped it. But point precedes the mark in both the
             * previous and current selections.
             */

            /* dehilite the line we were on */
        APPDrawContext( pSrtRng.fMark.fCol->GetAPPName( ), pSrtRng.fMark.fCol, drawCtx );
        pSrtRng.fMark.fTxl->Hilite( &pSrtRng.fMark, pSrtRng.fMark.fHLoc,
                                    NULL, LONG_MAX,
                                    drawCtx, func, pSrtRng );

            /* Start where old range left off.  */
        cSrtRng.fMark       = prevRange.fPoint;

            /* Extend to end of this first line.*/
        cSrtRng.fMark.fHLoc = LONG_MAX;

            /* Point is unchanged.              */
        cSrtRng.fPoint      = fPoint;
```

```
                        txl->Hilite( NULL, LONG_MIN, NULL, LONG_MAX, drawCtx, func, cSrtRng );
                    }
                }
            }
        }
    }

/* ==================================================================== */
/* mark all the lines of a selection range with the passed in bit */

void scSelection::MarkLayoutBits( const scLayBits& mark )
{
    scColumn*   c1;
    scColumn*   c2;
    scColumn*   c;
    scTextline* t1;
    scTextline* t2;
    scTextline* t;

    scSelection cSrtRng( *this );       /* current sorted range */
    cSrtRng.Sort( );


    c1 = cSrtRng.fMark.fCol;
    c2 = cSrtRng.fPoint.fCol;
    t1 = cSrtRng.fMark.fTxl;
    t2 = cSrtRng.fPoint.fTxl;

    if ( t1 == t2 )
        t1->Mark( mark );
    else if ( c1 == c2 ) {
        for ( ; t1 && t1 != LNNext(t2); t1 = LNNext(t1) )
            t1->Mark( mark );
    }
    else {
        for ( c = c1; c && c != c2->GetNext(); c = c->GetNext() ) {
            if ( c == c1 ) {
                t1->Mark( mark );
                for ( t = LNNext( t1 ); t; t = LNNext( t ) )
                    t->Mark( mark );
            }
            else if ( c == c2 ) {
                for ( t = c2->GetFirstline(); t && t != LNNext(t2); t = LNNext( t ) )
                    t->Mark( mark );
            }
            else {
                for ( t = c->GetFirstline(); t; t = LNNext( t ) )
                    t->Mark( mark );
            }
        }
    }
}

/* ==================================================================== */
/* called from a continue click or after a appmouse down,
 * hilite the new area
 */

void scSelection::InteractiveHilite( scSelection&   prevRange,
                                     HiliteFuncPtr  func )
{
    scSelection     cSrtRng( *this );       /* current sorted range     */
    scSelection     pSrtRng( prevRange );   /* previous sorted range    */
    APPDrwCtx       drawCtx;

    if ( pSrtRng == cSrtRng )
        return;

    pSrtRng.Sort( );
    cSrtRng.Sort( );

    if ( pSrtRng.fMark.fTxl == pSrtRng.fPoint.fTxl ) {
            // previous selection was one line
        if ( stream && lastPara ) {
            fMark.fPara         = stream->First();
            fMark.fOffset       = 0;
            fPoint.fPara        = lastPara;
            fPoint.fOffset      = PARAChSize( lastPara );
            fMark.fEndOfLine    = false;
            fPoint.fEndOfLine   = false;

            ChangeSelection( SLC_LOCATE, false );
        }
    }
}

/* ================================================================== */
/* hilite the selection range */

void scSelection::LineHilite( HiliteFuncPtr func )
{
    scColumn*   col1;
    scColumn*   col2;
    scTextline* txl1;
    scTextline* txl2;

    scColumn*   col;
    scTextline* txl;

    APPDrwCtx   drawCtx;

    scAssertValid();

    scSelection cSrtRng( *this );
    cSrtRng.Sort( );

    col1 = cSrtRng.fMark.fCol;
    col2 = cSrtRng.fPoint.fCol;
    txl1 = cSrtRng.fMark.fTxl;
    txl2 = cSrtRng.fPoint.fTxl;

#if SCDEBUG > 1
    cSrtRng.DbgPrintInfo( 3 );
#endif

    if ( col1 && col2 && txl1 && txl2 ) {

        if ( txl1 == txl2 ) {
            APPDrawContext( col1->GetAPPName(), col1, drawCtx );

            txl1->Hilite( &cSrtRng.fMark, cSrtRng.fMark.fHLoc, &cSrtRng.fPoint, cSrtRng.fPoint.fHLoc
, drawCtx, func, cSrtRng );
        }
        else if ( col1 == col2 ) {
            col1->Hilite( cSrtRng.fMark, cSrtRng.fPoint, func, cSrtRng );
        }
        else {
            for ( col = col1; col && col != col2->GetNext(); col = col->GetNext() ) {
                if ( col == col1 ) {
                    APPDrawContext( col1->GetAPPName(), col1, drawCtx );

                    txl1->Hilite( &cSrtRng.fMark, cSrtRng.fMark.fHLoc, NULL, LONG_MAX, drawCtx, func
, cSrtRng );
                    for ( txl = LNNext(txl1); txl; txl = LNNext( txl ) )
                        txl->Hilite( NULL, LONG_MIN, NULL, LONG_MAX, drawCtx, func, cSrtRng );
                }
                else if ( col == col2 ) {
                    APPDrawContext( col2->GetAPPName(), col2, drawCtx );
                    for ( txl = col2->GetFirstline(); txl && txl != txl2; txl = LNNext( txl ) )
                        txl->Hilite( NULL, LONG_MIN, NULL, LONG_MAX, drawCtx, func, cSrtRng );
                    txl2->Hilite( NULL, LONG_MIN, &cSrtRng.fPoint, cSrtRng.fPoint.fHLoc, drawCtx, fu
nc, cSrtRng );
                }
                else {
                    APPDrawContext( col->GetAPPName(), col, drawCtx );
                    for ( txl = col->GetFirstline(); txl; txl = LNNext( txl ) )
        fPoint.fOffset  = txl->GetEndOffset( );
        if (
#ifdef GermanHyphenation
            txl->IsHyphenated() && ! SLCNextLineChar( txlH )
#else
            txl->IsHyphenated()
#endif
        )
            fPoint.fEndOfLine = true;
        else
            fPoint.fEndOfLine = false;

        ChangeSelection( SLC_LOCATE, false );
    }
}

/* ================================================================== */
/* take the current selection and convert it into a para selection */

void scSelection::ParaSelect( void )
{
    scContUnit* para;

    ChangeSelection( SLC_LOCATE, false );
    para = fMark.fPara;

    if ( para ) {
        fMark.fOffset       = 0;
        fPoint.fOffset      = PARAChSize( para );
        fMark.fEndOfLine    = false;
        fPoint.fEndOfLine   = false;

        ChangeSelection( SLC_LOCATE, false );
    }
}

/* ================================================================== */
/* take the current selection and convert it into a column selection */

void scSelection::ColumnSelect( void )
{
    ChangeSelection( SLC_LOCATE, false );

    if ( fMark.fCol ) {
        scTextline* firstline   = fMark.fCol->GetFirstline();
        scTextline* lastline    = fMark.fCol->GetLastline();

        fMark.fPara             = firstline->GetPara();
        fMark.fOffset           = firstline->GetStartOffset();
        fMark.fEndOfLine        = false;

        fPoint.fPara                = lastline->GetPara();
        fPoint.fOffset          = lastline->GetEndOffset();
        fPoint.fEndOfLine       = true;

        ChangeSelection( SLC_LOCATE, false );
    }
}

/* ================================================================== */

/* take the current selection and convert it into a selection of the
 * entire stream
 */

void scSelection::AllSelect( void )
{
    scStream    *stream;
    scContUnit  *lastPara;

    if ( fMark.fCol ) {
        stream          = fMark.fCol->GetStream();
        lastPara        = stream->Last( );
```

```
/* ================================================================== */

int TextMarker::operator>=( const TextMarker& tm ) const
{
    if ( fParaCount > tm.fParaCount )
        return true;
    else if ( fParaCount == tm.fParaCount )
        return fOffset >= tm.fOffset;
    return false;
}

/* ================================================================== */

int TextMarker::operator<=( const TextMarker& tm ) const
{
    if ( fParaCount < tm.fParaCount )
        return true;
    else if ( fParaCount == tm.fParaCount )
        return fOffset <= tm.fOffset;
    return false;
}

/* ================================================================== */

int TextMarker::operator==( const TextMarker& tm ) const
{
    return fParaCount == tm.fParaCount && fOffset == tm.fOffset;
}

/* ================================================================== */

int TextMarker::operator!=( const TextMarker& tm ) const
{
    return fParaCount != tm.fParaCount || fOffset != tm.fOffset;
}

/* ================================================================== */

// sort the selection so that the mark appears before the point

void scSelection::Sort()
{
    if ( fMark <= fPoint )
        return;

    TextMarker temp( fMark );

    fMark   = fPoint;
    fPoint  = temp;
}

/* ================================================================== */
/* take the current selection and convert it into a word selection */

void scSelection::WordSelect( void )
{
    fMark.fPara->SelectWord( fMark.fOffset, fMark.fOffset, fPoint.fOffset );
    ChangeSelection( SLC_LOCATE, false );
}

/*==================================================================*/
/* take the current selection and convert it into a line selection */

void scSelection::LineSelect( void )
{
    scTextline* txl;

    ChangeSelection( SLC_LOCATE, false );
    txl = fMark.fTxl;

    if ( txl ) {
        fMark.fOffset   = txl->GetStartOffset( );
    return result;
}

#endif

/* ================================================================== */

#if SCDEBUG > 1

/* ================================================================== */

void TextMarker::DbgPrintInfo( int debugLevel ) const
{
    SCDebugTrace( debugLevel, scString( "fCol 0x%08x fPara 0x%08x fTxl 0x%08x\n" ),
                  fCol, fPara, fTxl );
    SCDebugTrace( debugLevel, scString( "fColCount %d fParaCount %d fLineCount %d fOffset %d\n" ),
                  fColCount, fParaCount, fLineCount, fOffset );
}

/* ================================================================== */

void TextMarker::scAssertValid()
{
    if ( fCol )
        fCol->scAssertValid();
    if ( fPara )
        fPara->scAssertValid();
    if ( fTxl )
        fTxl->scAssertValid();
}

/* ================================================================== */

void scSelection::scAssertValid()
{
    fMark.scAssertValid();
    fPoint.scAssertValid();
}

/* ================================================================== */

void scSelection::DbgPrintInfo( int debugLevel ) const
{
    SCDebugTrace( debugLevel, scString( "scSelection: \n" ) );
    fMark.DbgPrintInfo( debugLevel );
    fPoint.DbgPrintInfo( debugLevel );
}

/* ================================================================== */

#endif

/* ================================================================== */

int TextMarker::operator>( const TextMarker& tm ) const
{
    if ( fParaCount > tm.fParaCount )
        return true;
    else if ( fParaCount == tm.fParaCount )
        return fOffset > tm.fOffset;
    return false;
}

/* ================================================================== */

int TextMarker::operator<( const TextMarker& tm ) const
{
    if ( fParaCount < tm.fParaCount )
        return true;
    else if ( fParaCount == tm.fParaCount )
        return fOffset < tm.fOffset;
    return false;
}
```

```
        // the para and the offset are the only reliable values,
        // derive the others
    //scAssert( ggcS.scRecomposeActive != false );

    /* We use if-else instead of switch() because K&R requires */
    /* the switch() expression to be an ``int'', not a ``long''. */
    /* And oh yes, Microsoft C 6.0 generates buggy code for such a switch() */
    /* even though it claims to be ANSI-conformant! */

    if ( !arrowKey ) {
        if ( tmMove != SLC_LOCATE ) {
            fMark.fOffset   += tmMove;
            fPoint.fOffset  += tmMove;
        }
    }
    else {
        switch ( tmMove ) {
            case NEXT_LINE:
            case PREV_LINE:
                fMark.fOffset   = tmMove;
                fPoint.fOffset  = tmMove;
                break;
            case -1:
                if ( fMark.fTxl                                   &&
                     fMark.fOffset == fMark.fTxl->GetStartOffset( ) &&
                     ( ( prevLine = LNPrev( fMark.fTxl ) ) != 0 )   &&
                     prevLine->IsHyphenated() ) {
                    fMark.fEndOfLine    = true;
                    fPoint.fEndOfLine   = true;
                }
                else {
                    fMark.fEndOfLine    = false;
                    fPoint.fEndOfLine   = false;
                    fMark.fOffset--;
                    fPoint.fOffset--;
                }
                break;
            case 1:
                if ( fMark.fEndOfLine ) {
                    fMark.fEndOfLine    = false;
                    fPoint.fEndOfLine   = false;
                }
                else {
                    fMark.fOffset++;
                    fPoint.fOffset++;
                    if ( fMark.fTxl                                   &&
                         fMark.fOffset == fMark.fTxl->GetEndOffset()    &&
                         fMark.fTxl->IsHyphenated() ) {
                        fMark.fEndOfLine    = true;
                        fPoint.fEndOfLine   = true;
                    }
                }
                break;
        }
    }
//  SCASSERT ( select->fMark.fPara == select->fPoint.fPara );
    fMark.Update( cursDirect, GetFlowset() );
    return fPoint.Update( cursDirect, GetFlowset() );
}

/* ================================================================== */
/* Returns true if a German hyphenation spelling change also affects    */
/* the following text line.                                             */

#ifdef GermanHyphenation

static Bool SLCNextLineChar( scTextline* txl )
{
    Bool        result;

    result  = txl->fNextLineChar;
```

```
        locate = false;
    }
    else if ( fOffset < 0 ) {
        /* previous paragraph */
        if ( fPara->GetPrev() ) {
            fPara   = fPara->GetPrev( );
            fOffset = PARAChSize( fPara );
        }
        else {
            fOffset      = 0;
            selectionMoved  = false;
        }
        fEndOfLine  = false;
    }
    else if ( fOffset > PARAChSize( fPara ) ) {
        /* next paragraph */
        if ( fPara->GetNext( ) ) {
            fPara   = fPara->GetNext( );
            fOffset = 0;
        }
        else {
            fOffset      = PARAChSize( fPara );
            selectionMoved  = false;
        }
        fEndOfLine  = false;
    }

    fParaCount  = fPara->GetCount();

        // PARAFindLocation will set fEndOfLine to false if it
        // finds it is no longer relevant (if we are in the middle
        // of a line, for example).
    if ( locate )
        fPara->FindLocation( fOffset, fEndOfLine, fTxl, fHLoc, movement );

    if ( fTxl ) {
        fCol = col  = fTxl->GetColumn();
        fColCount   = col->GetCount();
        fLineCount  = fTxl->GetLinecount();
        if ( setPosition )
            fSelMaxX = fHLoc;
    }
    else
        Zero( thePOINT, false );

    return selectionMoved;

/* ==================================================================== */
/* something has happened to change the selection, let us correct it */

void scSelection::UpdateSelection()
{
    if ( fMark.fPara )
        fMark.Update( eCursNoMovement, fFlowset );
    if ( fPoint.fPara )
        fPoint.Update( eCursNoMovement, fFlowset );
}

/* ==================================================================== */

void scSelection::CheckFreePara( scContUnit* p )
{
}

/* ==================================================================== */

Bool scSelection::ChangeSelection( long tmMove,
                                   Bool arrowKey )
{
    scTextline*         prevLine;
    eContentMovement    cursDirect = tmMove > 0 ? eCursForward : eCursBackward;
```

```
                // it will select it - otherwise it will simply fail and
                // zero the text marker
            if ( fOffset < 0 )
                fOffset = 0;

                // PARAFindLocation will set fEndOfLine to false if it
                // finds it is no longer relevant (if we are in the middle
                // of a line, for example).
            if ( !fPara->FindLocation( fOffset, fEndOfLine, fTxl, fHLoc, movement ) ) {
                if ( fPara && PARAChSize( fPara ) < fOffset )
                    fOffset = PARAChSize( fPara );
                Zero( thePOINT, false );
                return false;
            }
            else {
                fCol = col   = fTxl->GetColumn();
                fColCount    = col->GetCount( );
                fLineCount   = fTxl->GetLinecount();
                fSelMaxX          = fHLoc;
                return selectionMoved;
            }
        }

        if ( fOffset == NEXT_LINE || fOffset == PREV_LINE ) {
            if ( fOffset == NEXT_LINE )
                txl = fTxl->GetNextLogical();
            else
                txl = fTxl->GetPrevLogical();

            if ( txl ) {
                fCol = txl->GetColumn();
#if 0
                if ( fCol->GetFlowdir().IsVertical() ) {
                    if ( txl->GetOrigin().y + txl->GetMeasure() <= fSelMaxX && txl->IsHyphenated() )
                        fEndOfLine = true;
                    else
                        fEndOfLine = false;
                }
                else {
                    if ( txl->GetOrigin().x + txl->GetMeasure() <= fSelMaxX && txl->IsHyphenated() )

                        fEndOfLine = true;
                    else
                        fEndOfLine = false;
                }
#endif
                fTxl = txl;
            }
            else
                selectionMoved = false;

            fPara    = fTxl->GetPara( );
            setPosition = false;

            flowDir = fTxl->GetFlowdir();
            vertical = flowDir.IsVertical();

            mPt      = fTxl->GetOrigin();
            if ( vertical ) {
                mPt.y    = fSelMaxX;
                mPt.Translate( -fTxl->GetVJOffset(), 0 );
            }
            else {
                mPt.x    = fSelMaxX;
                mPt.Translate( 0, fTxl->GetVJOffset() );
            }

            scMuPoint   charOrg;
            fTxl->Select( charOrg, fOffset, mPt, movement, fEndOfLine );
            if ( vertical )
                fHLoc = charOrg.y;
            else
                fHLoc = charOrg.x;
```

```
/* zero out the layout portions of the text marker, the logical selection
 * remains the same but something has changed the layout structure and we
 * need to zero out that portion of the selection
 */

void TextMarker::Zero( int  type,
                       Bool zeroLogical )
{
    fCol        = 0;
    fTxl        = 0;
    fEndOfLine  = false;

    switch ( type ) {
        default:
        case thePOINT:
            fSelMaxX    = fHLoc         = LONG_MAX;
            fColCount   = fLineCount    = LONG_MAX;
            break;
        case theMARK:
            fSelMaxX        = fHLoc     = LONG_MIN;
            fColCount   = fLineCount    = LONG_MIN;
            break;
    }

    if ( zeroLogical ) {
        fPara       = 0;
        fParaCount  = 0;
        fOffset     = 0;
    }
}

/* ==================================================================== */

void TextMarker::UpdateInfo( int setMax )
{
    if ( fPara ) {
        fPara->FindLocation( fOffset, fEndOfLine, fTxl, fHLoc, eCursNoMovement );

        if ( fTxl ) {
            fCol        = fTxl->GetColumn();
            fColCount   = fCol->GetCount();
            fLineCount  = fTxl->GetLinecount();
            if ( setMax )
                fSelMaxX = fHLoc;
        }
    }
    else
        Invalidate();

}

/* ==================================================================== */
// move the selection based upon the value of tmMove

Bool TextMarker::Update( eContentMovement   movement,
                         scColumn*          flowset )
{
    scColumn*   col;
    scTextline* txl;
    scMuPoint   mPt;
    Bool        setPosition     = true;
    Bool        locate          = true;
    Bool        selectionMoved  = true;
    Bool        vertical        = false;
    scFlowDir   flowDir;

    if ( fTxl == 0 ) {
        fParaCount  = fPara->GetCount();

        fEndOfLine  = false;

            // this indicates previous line, lets set it to the beginning
            // of the paragraph and see what happens, if it finds a line
    fMark.Invalidate();
    fPoint.Invalidate();
}

/* ==================================================================== */
/* insure that we have a valid selection */

Bool scSelection::ValidateSelection( scSelection& validSelect ) const
{
    scColumn*   col1;
    scColumn*   col2;
    scContUnit* lastPara;
    scTextline* lastTxl;
    scFlowDir   flowDir;

    col1 = fMark.fCol;
    col2 = fPoint.fCol;
    if ( col1 && col2 )
        validSelect = *this;
    else if ( col1 ) {
            // the mark lies outside the visible region

            // establish the last visible location that is visible
        lastPara             = fMark.fPara->GetLastVisiblePara();
        lastTxl              = lastPara->GetLastVisibleLine();

        col2                 = lastTxl->GetColumn();
        flowDir              = col2->GetFlowdir();

            // mark should be ok
        validSelect.fMark            = fMark;

            // set up last visible point
        validSelect.fPoint.fPara         = lastPara;
        validSelect.fPoint.fParaCount    = lastPara->GetCount();

        validSelect.fPoint.fTxl          = lastTxl;
        validSelect.fPoint.fLineCount    = lastTxl->GetLinecount();
        validSelect.fPoint.fOffset       = lastTxl->GetEndOffset( );

        validSelect.fPoint.fCol          = col2;
        validSelect.fPoint.fColCount     = col2->GetCount( );


        if ( lastTxl->IsHyphenated() )
            validSelect.fPoint.fEndOfLine    = true;
        else
            validSelect.fPoint.fEndOfLine    = false;

#if 1
        validSelect.fPoint.fHLoc = LONG_MAX;
#else
        scXRect     xRect;
        lastTxl->QueryExtents( xRect );

        if ( flowDir.IsVertical() )
            validSelect.fPoint.fHLoc     = xRect.y2;
        else
            validSelect.fPoint.fHLoc     = xRect.x2;
#endif
    }
    else {
        validSelect.fMark.Zero( theMARK, true );
        validSelect.fPoint.Zero( thePOINT, true );
        return false;
    }

    validSelect.scAssertValid();

    return true;
}

/* ==================================================================== */
```

```
        TypeSpec ts = p->GetDefaultSpec();
        tsList.Insert( ts );
    }
}

/* ================================================================== */

void scSelection::GetCharSpecList( scSpecLocList& csList )
{
    long        offset1,
                offset2;
    scContUnit* p;
    scContUnit* p1;
    scContUnit* p2;
    scContUnit* lastPara;

    scSelection aRng( *this );
    aRng.Sort( );

    p1      = aRng.fMark.fPara;
    offset1 = aRng.fMark.fOffset;
    p2      = aRng.fPoint.fPara;
    offset2 = aRng.fPoint.fOffset;

    if ( p1 == p2 )
        p1->OffsetGetCharSpecList( offset1, offset2, csList );
    else {
        lastPara = p2->GetNext( );
        for ( p = p1; p && p != lastPara; p = p->GetNext() ) {
            if ( p == p1 )
                p->OffsetGetCharSpecList( offset1, LONG_MAX, csList );
            else if ( p == p2 )
                p->OffsetGetCharSpecList( 0, offset2, csList );
            else
                p->OffsetGetCharSpecList( 0, LONG_MAX, csList );
        }
    }
}

/* ================================================================== */

void scSelection::MarkValidatedSelection( const scLayBits& mark )
{
    scSelection     validatedSelection;

    if ( ValidateSelection( validatedSelection ) )
        validatedSelection.MarkLayoutBits( mark );
}

/* ================================================================== */

void scSelection::ValidateHilite( HiliteFuncPtr func ) const
{
    scSelection validatedSelection;

#if SCDEBUG > 1
    DbgPrintInfo( 3 );

    static int noHilite;

    if ( noHilite )
        return;
#endif

    if ( ValidateSelection( validatedSelection ) )
        validatedSelection.LineHilite( func );
}

/* ================================================================== */
/* invalidate the selection if the selection lies in the indicated stream */

void scSelection::Invalidate()
{
    offset2 = aRng.fPoint.fOffset;

    if ( para1 == para2 ) {
            // only one paragraph

        if (offset1 == offset2)
        {
            TypeSpec ts = para1->SpecAtOffset( offset1 );
            tsList.Insert( ts );
        }
        else
            para1->OffsetGetTSList( offset1, offset2, tsList );
    }
    else {
        lastPara = para2->GetNext( );
        for ( para = para1; para && para != lastPara; para = para->GetNext() ) {
            if ( para == para1 )
                para->OffsetGetTSList( offset1, LONG_MAX, tsList );
            else if ( para == para2 )
                para->OffsetGetTSList( 0, offset2, tsList );
            else
                para->OffsetGetTSList( 0, LONG_MAX, tsList );
        }
    }
}

/* ==================================================================== */

void scSelection::GetParaSpecList( scSpecLocList& csList )
{
    long        offset1,
                offset2;
    scContUnit* p;
    scContUnit* p1;
    scContUnit* p2;
    scContUnit* lastPara;

    scSelection aRng( *this );
    aRng.Sort( );

    p1      = aRng.fMark.fPara;
    offset1 = aRng.fMark.fOffset;
    p2      = aRng.fPoint.fPara;
    offset2 = aRng.fPoint.fOffset;

    lastPara = p2->GetNext( );
    for ( p = p1; p && p != lastPara; p = p->GetNext() ) {
        scSpecLocation specLoc( p->GetCount(), -1, p->GetDefaultSpec() );
        csList.Append( specLoc );
    }
}

/* ==================================================================== */

void scSelection::GetParaSpecList( scTypeSpecList& tsList )
{
    long        offset1,
                offset2;
    scContUnit* p;
    scContUnit* p1;
    scContUnit* p2;
    scContUnit* lastPara;

    scSelection aRng( *this );
    aRng.Sort( );

    p1      = aRng.fMark.fPara;
    offset1 = aRng.fMark.fOffset;
    p2      = aRng.fPoint.fPara;
    offset2 = aRng.fPoint.fOffset;

    lastPara = p2->GetNext( );
    for ( p = p1; p && p != lastPara; p = p->GetNext() ) {
```

```
/***************************************************************************

    File:      SCSELECT.C


    $Header: /Projects/Toolbox/ct/Scselect.cpp 2     5/30/97 8:45a Wmanis $

    Contains:  handles the selection object/id

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

***************************************************************************/

#include "scselect.h"
#include "scpubobj.h"
#include "sccolumn.h"
#include "scglobda.h"
#include "scparagr.h"
#include "scstream.h"
#include "sctextli.h"
#include "scexcept.h"
#include "sccallbk.h"
#include "scstcach.h"

#include <limits.h>

typedef enum ePointAndMark {
    thePOINT = 1,
    theMARK
};

#ifndef HUGE_VAL
#define HUGE_VAL    HUGE
#endif

/* ================================================================== */


#ifdef GermanHyphenation
static Bool     SLCNextLineChar( scTextline * );
#endif


#define SLC_LOCATE              (LONG_MAX-1)

/* ================================================================== */
/* get a list of type specs used in the selection */

void scSelection::GetTSList( scTypeSpecList& tsList )
{
    long        offset1,
                offset2;
    scContUnit* para;
    scContUnit* para1;
    scContUnit* para2;
    scContUnit* lastPara;

    scSelection aRng( *this );
    aRng.Sort( );       /* sort the selection */

    para1   = aRng.fMark.fPara;
    offset1 = aRng.fMark.fOffset;
    para2   = aRng.fPoint.fPara;
    int             StartWord( void );
    int             EndWord( void );

    int             PrevLine( void );
    int             NextLine( void );

    int             PrevEntireLine( void );
    int             NextEntireLine( void );

    int             StartLine( void );
    int             EndLine( void );

    int             PrevColumn( void );
    int             NextColumn( void );

    int             EndColumn( void );
    int             StartColumn( void );

    int             BeginPara( void );
    int             EndPara( void );

    int             Para( eSelectMove moveSelect );

    scColumn*       fFlowset;
    TextMarker      fMark;
    TextMarker      fPoint;

protected:

};

/***********************************************************************/
/***********************************************************************/


#endif /* _H_SCSELECT */
```

```
#ifdef _RUBI_SUPPORT
    Bool            GetAnnotation( int nth, scAnnotation& );
    void            ApplyAnnotation( const scAnnotation& , scRedispList* );
#endif

    scStream*       GetStream( void ) const;

    TypeSpec        GetSpecAtStart( void ) const;

                        // this will report the position of the point
                        // in relationship to the mark
    eSelectMovement RangeMovement( void ) const;


    void            Decompose( scStreamLocation&, scStreamLocation& );
    void            Decompose2( scStreamLocation&, scStreamLocation& );
    void            Restore( const scStreamLocation*,
                             const scStreamLocation*,
                             const scStream*,
                             Bool );


    scSelection&    operator=( const scSelection& sel )
                        {
                            fFlowset    = sel.fFlowset;
                            fMark       = sel.fMark;
                            fPoint      = sel.fPoint;
                            return *this;
                        }

    int             operator==( const scSelection& sel ) const
                        {
                            return (
                                fMark.fPara         == sel.fMark.fPara          &&
                                fMark.fParaCount    == sel.fMark.fParaCount     &&
                                fMark.fOffset       == sel.fMark.fOffset        &&
                                fMark.fEndOfLine    == sel.fMark.fEndOfLine     &&
                                fPoint.fPara        == sel.fPoint.fPara         &&
                                fPoint.fParaCount   == sel.fPoint.fParaCount    &&
                                fPoint.fOffset      == sel.fPoint.fOffset       &&
                                fPoint.fEndOfLine   == sel.fPoint.fEndOfLine
                            );
                        }


                        // invalidate the selection
    void            Invalidate( void );

    void            MoveSelect( eSelectMove );
    void            Extend( eSelectMove );
    void            WordSelect( void );
    void            LineSelect( void );
    void            ParaSelect( void );
    void            ColumnSelect( void );
    void            AllSelect( void );

                        // mark all the layout objects associated with
                        // the selection
    void            MarkLayoutBits( const scLayBits& );

#if SCDEBUG > 1
    void            DbgPrintInfo( int debugLevel = 0 ) const;
    void            scAssertValid( void );
#else
    void            scAssertValid( void ){}
#endif

    int             PrevWord( Scope scope = inStream );
    int             NextWord( Scope scope = inStream  );
    int             PrevSpellWord( Scope scope = inStream );
    int             NextSpellWord( Scope scope = inStream );
    void            NthPara( scStream* stream, long nthPara );
    void            SetParaSelection( scContUnit*   p,
                                      long          start,
                                      long          end );


    void            GetContUnits( scContUnit*& mark, scContUnit*& point ) const;

    void            CutText( scScrapPtr&, scRedispList* );
    void            ClearText( scRedispList*, Bool );
    void            CopyText( scScrapPtr& );
    void            PasteText( const scStream*, TypeSpec, scRedispList* );

    void            CopyAPPText( stTextImportExport& );
    void            PasteAPPText( stTextImportExport&, scRedispList* );

    void            InsertField( const clField&,
                                 TypeSpec&,
                                 scRedispList* );

    void            SetStyle( TypeSpec, scRedispList* );
    void            GetTSList( scTypeSpecList& );
    void            GetCharSpecList( scSpecLocList& );
    void            GetParaSpecList( scSpecLocList& );
    void            GetParaSpecList( scTypeSpecList& );


    void            SetFlowset( scColumn* col )         { fFlowset = col; }
    scColumn*       GetFlowset( void ) const            { return fFlowset; }

    void            SetMark( const TextMarker& mark )   { fMark = mark; }
    TextMarker&     GetMark( void )                     { return fMark; }

    void            SetPoint( const TextMarker& point ) { fPoint = point; }
    TextMarker&     GetPoint( void )                    { return fPoint; }


                        // is the selection just a caret (i.e. no content selected )
    Bool            IsSliverCursor( void ) const        { return fPoint == fMark; }
    int32           ContentSize() const;

                        // we are freeing this paragraph, check to see if we need
                        // to invalidate the selection
    void            CheckFreePara( scContUnit* );

    void            UpdateSelection( void );


                        // sort so that the mark occurs before the point
    void            Sort( void );

    void            ValidateHilite( HiliteFuncPtr ) const;
    Bool            ValidateSelection( scSelection& ) const;

    Bool            ChangeSelection( long   tmMove,
                                     Bool   arrowKey );

    void            CorrectSelection( scContUnit*   para1,
                                      long          ,
                                      scContUnit*   para2,
                                      long          offset2 );

    void            InteractiveHilite( scSelection&, HiliteFuncPtr );

    void            MarkValidatedSelection( const scLayBits& mark );

    void            LineHilite( HiliteFuncPtr );

    void            Iter( SubstituteFunc func, scRedispList* );
    void            KeyArray( short, scKeyRecord*, scRedispList* );
    void            TextTrans( eChTranType, int, scRedispList* );
    Bool            Update( eContentMovement, scColumn* flowset );


    void            UpdateInfo( int setMax );

    int             SelectPrevCharInPara( void );
    int             SelectNextCharInPara( void );

    int             SelectPrevChar( void );
    int             SelectNextChar( void );

    int             SelectPrevLine( void );
    int             SelectNextLine( void );

    int             SelectStartLine( void );
    int             SelectEndLine( void );

    Bool            SelectPrevWord( void  );
    Bool            SelectNextWord( void );

    Bool            SelectPrevSpellWord( void );
    Bool            SelectNextSpellWord( void );

    int             SelectStartWord( void );
    int             SelectEndWord( void );

    int             SelectStartSpellWord( int eleminateLeadingSpaces = 0 );
    int             SelectEndSpellWord( void );

    int             SelectStartPara( void );
    int             SelectEndPara( void );

    int             SelectPrevPara( void );
    int             SelectNextPara( void );

    int             SelectStartColumn( void );
    int             SelectEndColumn( void );

    int             SelectStartStream();
    int             SelectEndStream();

#if SCDEBUG > 1
    void            DbgPrintInfo( int debugLevel = 0 ) const;
    void            scAssertValid( void );
#else
    void            scAssertValid( void ){}
#endif
};


/* ================================================================== */
/* ================================================================== */
/* ================================================================== */

class scSelection : public scObject {
public:
    enum Scope {
        inContUnit,
        inStream,
        inLine,
        inColumn
    };
                    scSelection( scColumn* flowset = 0 ) :
                        fFlowset( flowset ){}

                    scSelection( const scSelection& sel )
                        {
                            fFlowset    = sel.fFlowset;
                            fMark       = sel.fMark;
                            fPoint      = sel.fPoint;
                        }
                        fPara( cu ),
                        fTxl( 0 ),
                        fColCount( 0 ),
                        fParaCount( poffset ),
                        fLineCount( 0 ),
                        fOffset( offset ),
                        fHLoc( 0 ),
                        fSelMaxX( 0 ),
                        fEndOfLine( false ) {}

                    TextMarker( const TextMarker& tm )
                        {
                            fCol        = tm.fCol;
                            fPara       = tm.fPara;
                            fTxl        = tm.fTxl;
                            fColCount   = tm.fColCount;
                            fParaCount  = tm.fParaCount;
                            fLineCount  = tm.fLineCount;
                            fOffset     = tm.fOffset;
                            fHLoc       = tm.fHLoc;
                            fSelMaxX    = tm.fSelMaxX;
                            fEndOfLine  = tm.fEndOfLine;
                        }


    scColumn*       fCol;
    scContUnit*     fPara;
    scTextline*     fTxl;
    long            fColCount;
    long            fParaCount;
    long            fLineCount;
    long            fOffset;
    MicroPoint      fHLoc;
    MicroPoint      fSelMaxX;
    Bool            fEndOfLine;

    TextMarker&     operator=( const TextMarker& tm )
                             {
                                fCol        = tm.fCol;
                                fPara       = tm.fPara;
                                fTxl        = tm.fTxl;
                                fColCount   = tm.fColCount;
                                fParaCount  = tm.fParaCount;
                                fLineCount  = tm.fLineCount;
                                fOffset     = tm.fOffset;
                                fHLoc       = tm.fHLoc;
                                fSelMaxX    = tm.fSelMaxX;
                                fEndOfLine  = tm.fEndOfLine;
                                return *this;
                             }

    int             operator>( const TextMarker& ) const;
    int             operator<( const TextMarker& ) const;
    int             operator>=( const TextMarker& ) const;
    int             operator<=( const TextMarker& ) const;
    int             operator==( const TextMarker& ) const;
    int             operator!=( const TextMarker& ) const;

    void            Zero( int, Bool );
    void            Invalidate( void )
                             {
                                fCol        = 0;
                                fPara       = 0;
                                fTxl        = 0;
                                fColCount   = -1;
                                fParaCount  = -1;
                                fLineCount  = -1;
                                fOffset     = -1;
                                fHLoc       = kInvalMP;
                                fSelMaxX    = kInvalMP;
                                fEndOfLine  = 0;
                             }
```

```
/***************************************************************************

    File:       SCSELECT.H


    $Header: /Projects/Toolbox/ct/Scselect.h 2     5/30/97 8:45a Wmanis $

    Contains:   scSelection definition

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

***************************************************************************/

#ifndef _H_SCSELECT
#define _H_SCSELECT

#ifdef SCMACINTOSH
#pragma once
#endif

#include "sccharex.h"
#include "scexcept.h"
#include "sctbobj.h"

class scColumn;
class scContUnit;
class scTextline;
class scAnnotation;
class scTypeSpecList;
class scSpecLocList;
class scRedispList;
class stTextImportExport;
class clField;

typedef enum eSelectMovements {
    eBeforeSelect = 1,
    eEqualSelect,
    eAfterSelect
} eSelectMovement;



 /* ================================================================== */
 /* ================================================================== */
 /* ================================================================== */

 class TextMarker {
 public:
                    TextMarker() :
                        fCol( 0 ),
                        fPara( 0 ),
                        fTxl( 0 ),
                        fColCount( 0 ),
                        fParaCount( 0 ),
                        fLineCount( 0 ),
                        fOffset( 0 ),
                        fHLoc( 0 ),
                        fSelMaxX( 0 ),
                        fEndOfLine( false ) {}

                    TextMarker( scContUnit* cu, int32 poffset, int32 offset ) :
                        fCol( 0 ),
    while ( fNumItems > 0 && ( ptr = Get( fNumItems - 1 ) ) == 0 )
        fNumItems--;

    ShrinkSlots();
}

/* ==================================================================== */

void scSet::Set( long index, const scObject* obj )
{
    if ( fElemSlots <= index )
        SetNumSlots( ( ( index / fBlockSize ) + 1 ) * fBlockSize );

    fNumItems = MAX( index + 1, fNumItems );

    scObject**  arr = (scObject**)fItems;

    scAssert( arr[index] == 0 );

    arr[index] = (scObject*)obj;
}

/* ==================================================================== */

void scSet::DeleteAll( void )
{
    int          i;
    scObject**   arr = (scObject**)fItems;
    scObject*    obj;

    for ( i = 0; i < fNumItems; i++ ) {
        obj     = *arr;
        *arr++  = 0;

        delete obj;
    }
    RemoveAll();
}

/* ==================================================================== */
```

```
/*==================================================================

    File:       CBAG.C

    $Header: /Projects/Toolbox/ct/SCSET.CPP 2     5/30/97 8:45a Wmanis $

    Contains:   Implementation of a set of objects.

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.


==================================================================*/

#include "scset.h"


/* ================================================================== */

scSet::scSet( const scSet& set ) :
                scMemArray( sizeof( void* ), true )
{
    scAssert( fNumItems == 0 );
    AppendData( set.GetMem(), set.GetNumItems() );
    scAssert( fNumItems == set.GetNumItems() );
}

/* ================================================================== */

long scSet::Index( const scObject* ptr ) const
{
    long                i;
    const scObject**    arr = (const scObject**)fItems;

    for ( i = 0; i < fNumItems; i++ )
        if ( *arr++ == ptr )
            return i;

    return -1;
}

/* ================================================================== */

long    scSet::Add( const scObject* obj )
{
    long index = Index( obj );

    if ( index >= 0 )
        return index;

    AppendData( (ElementPtr)&obj, 1 );
    return GetNumItems() - 1;
}

/* ================================================================== */

void scSet::Remove( const scObject* obj )
{
    long        index = Index( obj );
    scObject*   ptr;

    if ( index >= 0 )
        *( ( (scObject **)fItems) + index ) = 0;
```

```
#endif
```

```
/*===================================================================

    File:       scset.h

    $Header: /Projects/Toolbox/ct/SCSET.H 2     5/30/97 8:45a Wmanis $

    Contains:   Set definition.

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

@doc

===================================================================*/



#ifndef _H_SCBAG
#define _H_SCBAG

#ifdef SCMACINTOSH
#pragma once
#endif

#include "scmemarr.h"

/* ================================================================== */
/* ================================================================== */
// @class scSet contains an array of 32 bit pointers. Insertion appends
// the pointer and deletion zeros out the slot. This way we may
// insure that all inserted pointers will get a unique index and a valid
// pointer will never get the index zero.

class scSet : public scMemArray {
public:
                scSet() :
                    scMemArray( sizeof( scObject* ), true ){}

                scSet( const scSet& );

    scObject*   Get( long offset ) const
                    {
                        scAssert( offset < fNumItems );
                        return ((scObject**)fItems)[offset];
                    }

    Bool        Includes( const scObject* obj ) const
                    {
                        return Index( obj ) >= 0;
                    }

    long        Add( const scObject* );
    void        Remove( const scObject* );
    long        Index( const scObject* ) const;
    void        Set( long index, const scObject* );

                    // delete all the objects in the set and set
                    // the set to zero elements
    void        DeleteAll( void );
};
```

```
/*****************************************************************

    File:       SCSETJMP.H

    $Header: /Projects/Toolbox/ct/SCSETJMP.H 2     5/30/97 8:45a Wmanis $

    Contains:   take care of setjmp/longjmp

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

*****************************************************************/

#ifndef _H_SCSETJMP
#define _H_SCSETJMP

#if defined( SCMACINTOSH )

    #include <setjmp.h>

    #define scJMPBUF                    jmp_buf
    #define scTHROW( lpbuf, numback)    longjmp( lpbuf, numback )
    #define scCATCH( lpbuf )            setjmp( lpbuf )

#elif defined( SCWINDOWS )

    #include <windows.h>
    #include <setjmp.h>

    #ifdef _JMP_BUF_DEFINED
        #define scJMPBUF                    jmp_buf
        #define scTHROW( lpbuf, numback)    longjmp( lpbuf, numback )
        #define scCATCH( lpbuf )            setjmp( lpbuf )
    #else
        #define scJMPBUF                    CATCHBUF
        #define scTHROW( lpbuf, numback)    Throw( lpbuf, numback )
        #define scCATCH( lpbuf )            Catch( lpbuf )
    #endif

#endif


#endif /* _H_SCSETJMP */
```

```
            raise_if( size != 8, scERRfile );
        }
    }

}

/* ================================================================== */

void scSpecRecord::restorePointer()
{
}

/* ================================================================== */

void scSpecRun::RestorePointers( void )
{
    for ( int i = 0; i < NumItems(); i++ )
        (*this)[i].restorePointer();
}

/* ================================================================== */

long scSpecRun::ExternalSize( void ) const
{
    return sizeof( scSpecRecord ) * NumItems() + sizeof( int32 );
}

/* ================================================================== */

void scSpecRun::SetContentSize( int32 size )
{
    int index = NumItems() - 2;

    //PrintRun( "SetContentSize" );

    while ( index > 0 && size <= (*this)[index].offset() )
        Remove( index-- );

    //PrintRun( "SetContentSize" );

    DebugRun( "SetContentSize" );
}

/* ================================================================== */
```

```
        const scSpecRecord& s1 = (*this)[i];
        const scSpecRecord& s2 = (*this)[i+1];
        if ( (*this)[i].spec() == (*this)[i+1].spec() )
            Remove( i + 1 );
        else if ( (*this)[i].offset() >= (*this)[i+1].offset() ) {
            if ( outer )
                Remove( i + 1 );
            else
                Remove( i );
        }
        else
            i++;
    }
}

/* ==================================================================== */

void scSpecRun::Read( APPCtxPtr ctxPtr, IOFuncPtr readFunc )
{
    uchar           sbuf[8];
    const uchar*    pbuf;
    long            numspecs;

    ReadLong( numspecs, ctxPtr, readFunc, kIntelOrder );

    SetNumSlots( numspecs + 1 );

    for ( int i = 0; i < numspecs; i++ ) {
        ReadBytes( sbuf, ctxPtr, readFunc, 8 );

        pbuf = sbuf;

        ulong specid;
        pbuf = BufGet_long( pbuf, specid, kIntelOrder );

        ulong offset;
        pbuf = BufGet_long( pbuf, offset, kIntelOrder );

        TypeSpec spec( (stSpec*)APPDiskIDToPointer( ctxPtr, specid, diskidTypespec ) );

        scSpecRecord rec( spec, offset );

        Insert( i, rec );
    }

    consolidate();

    DebugRun( "scSpecRun::Read" );
}

/* ==================================================================== */

void scSpecRun::Write( APPCtxPtr    ctxPtr,
                       IOFuncPtr    writeFunc )
{
    int             i;
    uchar           sbuf[8];
    uchar*          pbuf;

        // do not write out the terminator
    WriteLong( NumItems() - 1, ctxPtr, writeFunc, kIntelOrder );

    for ( i = 0; i < NumItems(); i++ ) {
        pbuf = sbuf;

        if ( !(*this)[i].isTerminator() ) {
            long    diskid = APPPointerToDiskID( ctxPtr,
                                                 (*this)[i].spec().ptr(),
                                                 diskidTypespec );
            pbuf = BufSet_long( pbuf, diskid, kIntelOrder );
            pbuf = BufSet_long( pbuf, (*this)[i].offset(), kIntelOrder );

            long size = (*writeFunc)( ctxPtr, sbuf, 8 );
    for ( index = 0; NumItems() > 1 && index+1 < NumItems();  ) {
        if ( (*this)[index].offset() >= (*this)[index+1].offset()
                || (*this)[index].spec() == (*this)[index+1].spec() ) {
            Remove( index + 1 );
        }
        else
            index++;
    }
}

/* ================================================================== */

#ifdef SCDEBUG

void scSpecRun::DebugRun( const char* str ) const
{
    int foo;

    if ( (*this)[0].offset() != 0 || !isTerminated() )
        PrintRun( str );

    scAssert( (*this)[0].offset() == 0 );
    isTerminated();

#if 1
    static int debugrun;

    if ( debugrun )
        PrintRun( str );
    else {
        int printit = false;

        for ( int index = 0; index < NumItems() - 1; index++ ) {
            const scSpecRecord& s1 = (*this)[index];
            const scSpecRecord& s2 = (*this)[index+1];
            if ( (*this)[index].spec() == (*this)[index+1].spec() )
                printit = true;
            if ( (*this)[index].offset() >= (*this)[index+1].offset() )
                printit = true;
        }

        if ( printit )
            PrintRun( str );
    }
#endif

    for ( int i = 0; i < NumItems() - 1; i++ ) {
        const scSpecRecord& s1 = (*this)[i];
        const scSpecRecord& s2 = (*this)[i+1];
        scAssert( (*this)[i].spec() != (*this)[i+1].spec() );
        scAssert( (*this)[i].offset() < (*this)[i+1].offset() );
        foo = i;
    }
}

#endif

/* ================================================================== */

void scSpecRun::PrintRun( const char* info ) const
{
    SCDebugTrace( 0, "RUN: \"%s\"\n", info );

    for ( int i = 0; i < NumItems(); i++ )
        SCDebugTrace( 0, "\t%d\t%11d\t0x%08x\n", i, (*this)[i].offset(), (*this)[i].spec().ptr() );
}

/* ================================================================== */

void scSpecRun::consolidate( int outer )
{
    for ( int i = 0; i < NumItems() - 1;  ) {
```

```
    RemoveAll();
    for ( int i = 0; i < sr.NumItems(); i++ )
        Append( sr[i] );
    return *this;
}

/* ================================================================= */

int scSpecRun::operator==(const scSpecRun& sr ) const
{
    if ( NumItems() != sr.NumItems() )
        return 0;
    for ( int i = 0; i < NumItems(); i++ ) {
        if ( (*this)[i] != sr[i] )
            return 0;
    }
    return 1;
}

/* ================================================================= */

int scSpecRun::operator!=(const scSpecRun& ) const
{
    return 0;
}

/* ================================================================= */

int scSpecRun::isTerminated() const
{
    int index = NumItems() - 1;

    if ( index >= 0 )
        return (*this)[ index ].offset() == terminate_;
    return 0;
}

/* ================================================================= */

void scSpecRun::terminate()
{
    TypeSpec ts(0);
    scSpecRecord sr( ts, terminate_ );
    Append( sr );
}

/* ================================================================= */

int scSpecRun::indexAtOffset( int32 offset ) const
{
    int index;

    for ( index = 0; index < NumItems() && offset >= (*this)[index+1].offset(); index++ ) {
        if ( (*this)[index].isTerminator() )
            break;
    }
    return index;
}

/* ================================================================= */

void scSpecRun::backwardCleanUpRun( int startIndex )
{
    int         index;

    if ( startIndex > 0 ) {
        if ( (*this)[startIndex].offset() <= (*this)[startIndex-1].offset() ) {
            (*this)[startIndex].offset() = (*this)[startIndex-1].offset();      //??
            Remove( startIndex - 1 );
        }
        else if ( (*this)[startIndex].spec() == (*this)[startIndex-1].spec() )
            Remove( startIndex );
    }
```

```
        else if ( (*this)[startIndex].offset() <= offset )
            startIndex++;
        for ( index = startIndex; !(*this)[index].isTerminator(); index++ )
            (*this)[index].bumpOffset( amount );

            // if we were decrementing then check whether we now have
            // two specs with same offset
        backwardCleanUpRun( startIndex );
    }


    DebugRun( "BumpOffset: end" );
}


/* ================================================================== */

void scSpecRun::removeIndicies( int32 startIndex, int32 endIndex )
{
    int recsToRemove = endIndex - startIndex;

    for ( int i = 0; i < recsToRemove; i++ )
        Remove( startIndex );
}

/* ================================================================== */

int scSpecRun::Includes( TypeSpec spec )
{
    for ( int i = 0; i < NumItems(); i++ ) {
        if ( spec == (*this)[i].spec() )
            return 1;
    }
    return 0;
}

/* ================================================================== */

scSpecRecord& scSpecRun::SpecRecAtOffset( int32 offset )
{
    return (*this)[ indexAtOffset( offset ) ];
}

/* ================================================================== */

const scSpecRecord& scSpecRun::SpecRecAtOffset( int32 offset ) const
{
    return (*this)[ indexAtOffset( offset ) ];
}

/* ================================================================== */

void scSpecRun::Copy( scSpecRun& dst, int32 start, int32 end ) const
{
    dst.RemoveAll();
    //PrintRun( "scSpecRun::Copy" );

    int index1 = indexAtOffset( start );
    int index2 = indexAtOffset( end > start ? end - 1 : end  );

    for ( int i = index1; i <= index2; i++ ) {
        dst.Insert( i - index1, (*this)[ i ] );
        dst[i - index1].bumpOffset( -start );
    }
    dst.terminate();

    dst.DebugRun( "Copy: destination" );
}

/* ================================================================== */

scSpecRun& scSpecRun::operator=( const scSpecRun& sr )
{
    DebugRun( "Insert: destination 1" );
    sr.DebugRun( "Insert: source 1a" );

    TypeSpec startspec = SpecAtOffset( offset );
    scSpecRecord endrec( startspec, offset + len );

    int index = indexAtOffset( offset );
    BumpOffset( offset, len );

    //PrintRun( "Insert: destination 2" );

    for ( int i = 0; i < sr.NumItems() && !sr[i].isTerminator(); i++ ) {
        TypeSpec ts = sr[i].spec();
        scSpecRecord rec( ts, sr[i].offset() + offset );
        Insert( index + i + 1, rec );
    }

    //PrintRun( "Insert: destination 3" );

    Insert( index + i + 1, endrec );

    //PrintRun( "Insert: destination 4" );

    while( !(*this)[index].isTerminator() ) {
        if ( (*this)[index].spec() == (*this)[index+1].spec() ) {
            Remove( index+1 );
        }
        else if ( (*this)[index].offset() >= (*this)[index+1].offset() ) {
            Remove( index );
        }
        else
            index++;
    }
    consolidate();

    //PrintRun( "Insert: destination 5" );
    DebugRun( "Insert: source 5" );
}

/* ================================================================== */

void scSpecRun::bumpIndicies( int32 startIndex, int32 amount )
{
    for ( int index = startIndex; !(*this)[index].isTerminator(); index++ )
        (*this)[index].bumpOffset( amount );

        // if we were decrementing then check whether we now have two specs with same offset
    if ( amount < 0 )
        backwardCleanUpRun( startIndex );
}

/* ================================================================== */

void scSpecRun::BumpOffset( int32 offset, int32 amount )
{
    long        startIndex;
    long        index;

    if ( amount == 0 )
        return;

    startIndex = indexAtOffset( offset );

    if ( amount > 0 ) {
        for ( index = startIndex; !(*this)[index].isTerminator(); index++ ) {
            if ( (*this)[index].offset() > offset )
                (*this)[index].bumpOffset( amount );
        }
    }
    else {
        if ( !startIndex && (*this)[startIndex].offset() != 0 )
            ;
```

```
/* ================================================================== */

void scSpecRun::Clear( int32 start, int32 end )
{
    if ( start == end )
        return;

    int tailInsert = 0;
    TypeSpec ts = SpecAtOffset( end );
    scSpecRecord endrec( ts, end );

    int32   startIndex  = indexAtOffset( start );
    if ( (*this)[startIndex].offset() != start )
        startIndex++;
    int32   endIndex    = indexAtOffset( end );
    if ( (*this)[endIndex].offset() != end )
        tailInsert = 1;

#if 1
    while ( (*this)[startIndex].offset() < end )
        Remove( startIndex );

    if ( startIndex ) {
        while ( !(*this)[startIndex].isTerminator() && (*this)[startIndex -1 ].spec() == (*this)[sta
rtIndex].spec() )
            Remove( startIndex );
    }

    if ( tailInsert )
        Insert( startIndex, endrec );

    BumpOffset( start, -(end - start) );

#else
    if ( startIndex != endIndex ) {
        while ( (*this)[startIndex].offset() < end )
            Remove( startIndex );

        if ( startIndex ) {
            while ( !(*this)[startIndex].isTerminator() && (*this)[startIndex -1 ].spec() == (*this)
[startIndex].spec() )
                Remove( startIndex );
        }

        if ( tailInsert )
            Insert( startIndex, endrec );
    }

    BumpOffset( start, -(end - start) );
#endif

    DebugRun( "Clear: end" );
}

/* ================================================================== */

int32 scSpecRun::removeOffsets( int32 start, int32 end )
{
    int32   startIndex  = indexAtOffset( start );
    if ( (*this)[startIndex].offset() != start )
        startIndex++;

    while ( (*this)[startIndex].offset() < end )
        Remove( startIndex );
    return startIndex;
}

/* ================================================================== */

void scSpecRun::InsertRun( int32 offset, int32 len, const scSpecRun& sr )
{
    if ( len == 0 )
        return;
```

```
TypeSpec scSpecRun::GetLastSpec( void )
{
    return (*this)[NumItems()].spec();
}

/* ================================================================== */

int32 scSpecRun::GetLastOffset( void )
{
    if ( NumItems() > 2 )
        return (*this)[ NumItems() - 2 ].offset();
    return 0;
}

/* ================================================================== */

void scSpecRun::insertSpecRec( const scSpecRecord& sr, int32 index )
{
    Insert( index, sr );
}

/* ================================================================== */

void scSpecRun::AppendSpec( TypeSpec ts, int32 offset )
{
    int index = NumItems() - 1;
    scAssert( index >= 0 );

    scSpecRecord rec( ts, offset );
    Insert( index, rec );
    consolidate( 0 );
}

/* ================================================================== */

void scSpecRun::ApplySpec( TypeSpec spec, int32 start, int32 end )
{
    int startIndex = indexAtOffset( start );
    int endIndex   = indexAtOffset( end );

    if ( spec == (*this)[startIndex].spec() ) {
        if ( startIndex == endIndex )
            return;
        if ( startIndex + 1 == endIndex && (*this)[endIndex].offset() == end )
            return;
    }

    TypeSpec ts = SpecAtOffset( end );
    scSpecRecord endrec( ts, end );

    if ( (*this)[startIndex].offset() == start )
        (*this)[startIndex].spec_.exch( spec );
    else {
        scSpecRecord startrec( spec, start );
        insertSpecRec( startrec, ++startIndex );
    }
    ++startIndex;
    while ( (*this)[startIndex].offset() <= end ) {
        if ( !(*this)[startIndex].isTerminator() )
            Remove( startIndex );
        else
            break;
    }


    if ( !endrec.isTerminator() )
        insertSpecRec( endrec, startIndex );

    consolidate();

    DebugRun( "ApplySpec - end" );
}
```

```
/*==================================================================

    File:       nspcrec.c

    $Header: /Projects/Toolbox/ct/Scspcrec.cpp 2     5/30/97 8:45a Wmanis $

    Contains:   xxx put contents here xxx

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

==================================================================*/


#include "scspcrec.h"
#include "scfileio.h"
#include "sccallbk.h"
#include "scexcept.h"

int32 scSpecRun::terminate_ = LONG_MAX;

scSpecRun::scSpecRun()
{
    terminate();
}

/* ================================================================== */

scSpecRun::scSpecRun( const scSpecRun& sr )
{
    for ( int i = 0; i < sr.NumItems(); i++ )
        Append( sr[i] );
}

/* ================================================================== */

scSpecRun::scSpecRun( TypeSpec firstspec )
{
    Append( scSpecRecord( firstspec, 0 ) );
    terminate();
}

/* ================================================================== */

scSpecRun::~scSpecRun()
{
}

/* ================================================================== */

TypeSpec scSpecRun::GetFirstSpec( void )
{
    return (*this)[0].spec();
}

/* ================================================================== */

TypeSpec scSpecRun::SpecAtOffset( int32 offset )
{
    return (*this)[ indexAtOffset( offset ) ].spec();
}

/* ================================================================== */
```

```
                    }
    TypeSpec    spec()
                    {
                        return spec_;
                    }
    int32           offset() const
                    {
                        return offset_;
                    }
    int32&          offset()
                    {
                        return offset_;
                    }


    void            restorePointer();

private:
    TypeSpec    spec_;      // the spec to apply
    int32       offset_;    // character offset to start appling spec
};


/*======================================================================*/

#endif /* _H_SCSPECRU */
```

09529370 . 073400

```
    void                PrintRun( const char* ) const;
#else
    void                DebugRun( const char* ) const {}
    void                PrintRun( const char* ) const {}
#endif

    void                Read( APPCtxPtr, IOFuncPtr );
    void                Write( APPCtxPtr, IOFuncPtr );
    void                RestorePointers( void );

    long                ExternalSize( void ) const;
    int                 indexAtOffset( int32 ) const;

    static int32        terminate_;

private:

    void                insertSpecRec( const scSpecRecord&, int32 );
    int32               removeOffsets( int32, int32 );
    void                removeIndicies( int32, int32 );
    void                bumpIndicies( int32, int32 );

    void                backwardCleanUpRun( int );

    void                consolidate( int outer = 1 );

    void                terminate();

};

class scSpecRecord {
friend class scSpecRun;
public:
                scSpecRecord() :
                        spec_(0),
                        offset_(scSpecRun::terminate_)
                        {
                        }

                scSpecRecord( TypeSpec& ts, int32 offset ) :
                            spec_(ts), offset_(offset)
                        {
                        }
                ~scSpecRecord()
                        {
                        }
    void        set( TypeSpec& ts, int32 offset )
                    {
                        spec_   = ts;
                        offset_ = offset;
                    }
    int         isTerminator( void ) const
                    {
                        return offset_ == scSpecRun::terminate_;
                    }
    void        bumpOffset( int32 bump )
                    {
                        offset_ += (offset_!=scSpecRun::terminate_ ? bump : 0);
                        offset_ = ( offset_ < 0 ) ? 0 : offset_;
                    }

    int         operator==(const scSpecRecord& sr ) const
                    {
                        return spec_==sr.spec_ && offset_==sr.offset_;
                    }
    int         operator!=(const scSpecRecord& sr ) const
                    {
                        return spec_!=sr.spec_ || offset_!=sr.offset_;
                    }

    TypeSpec    spec() const
                    {
                        return spec_;
```

```
/*================================================================

    File:       SCSPECRU.H

    $Header: /Projects/Toolbox/ct/SCSPCREC.H 2     5/30/97 8:45a Wmanis $

    Contains:   scSpecRecord - spec plus content unit offset

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

================================================================*/

#ifndef _H_SCSPCREC
#define _H_SCSPCREC

#include "sctypes.h"
#include "scarray.h"

#include <limits.h>

/*=====================================================================*/
class scSpecRecord;


class scSpecRun : public scSizeableArrayD<scSpecRecord> {
public:
                    scSpecRun();
                    scSpecRun( const scSpecRun& );
                    scSpecRun( TypeSpec firstspec );
                    ~scSpecRun();

        // get the last valid spec in the run
    TypeSpec    GetFirstSpec( void );
    TypeSpec    SpecAtOffset( int32 );
    TypeSpec    GetLastSpec( void );

        // get the offset of the last valid spec
    int32           GetLastOffset( void );

    int             Includes( TypeSpec );

    void            AppendSpec( TypeSpec, int32 );
    void            ApplySpec( TypeSpec, int32, int32 );
    void            Clear( int32, int32  );
    void            Copy( scSpecRun&, int32, int32 ) const;
    void            InsertRun( int32 offset, int32 len, const scSpecRun& );

    void            BumpOffset( int32, int32 );
    void            SetContentSize( int32 );

        // return the record of the spec rec at an offset
    scSpecRecord&       SpecRecAtOffset( int32 );
    const scSpecRecord& SpecRecAtOffset( int32 ) const;

    scSpecRun&      operator=( const scSpecRun& );
    int             operator==(const scSpecRun& ) const;
    int             operator!=(const scSpecRun& ) const;

    int             isTerminated() const;

 #ifdef SCDEBUG
    void            DebugRun( const char* ) const;
```

```
/* ================================================================== */
```

09620370.073100

```
) * kSmallCapCorrection );
                        else
                            rluWidth = FIgetRLUEscapement( fSpec, GetCorrectedGlyph( ch ) );

                        if ( rluWidth == scBaseRLUsystem ) {
                            fWidths[ch] = scRoundMP( conversion * rluWidth );
                            if ( fFlowDir.IsHorizontal() )
                                fWidths[ch] = GetSetSize() + GetOptLSP();
                            else
                                fWidths[ch] = GetPtSize() + GetOptLSP();
                        }
                        else
                            fWidths[ch] = scRoundMP( conversion * rluWidth );
                    }
                break;

                case 0:
                case scVertTab:
                case scHardReturn:
                    if ( fFlowDir.IsHorizontal() )
                        fWidths[ch] = GetSetSize();
                    else
                        fWidths[ch] = GetPtSize();
                    break;

                case scNoBreakSpace:
                    fWidths[ch] = GetOptWord();
                    break;

                    // these really need to be further up stream
                case scNoBreakHyph:
                case scDiscHyphen:
                    fWidths[ch] = 0;
                    break;
                case scFigureSpace:
                    fWidths[ch] = scRoundMP( conversion * FIgetRLUEscapement( fSpec, '0' ) );
                    break;
                case scThinSpace:
                    fWidths[ch] = scRoundMP( conversion * scBaseRLUsystem / 6 );
                    break;
                case scEnSpace:
                    fWidths[ch] = scRoundMP( conversion * scBaseRLUsystem / 2 );
                    break;
                case scEmSpace:
                    fWidths[ch] = scRoundMP( conversion * scBaseRLUsystem );
                    break;
            }
        }
        if ( fWidths[ch] == 0 )
            return 0;
    }
    return fWidths[ch] + GetOptLSP();
}

/* ==================================================================== */

GlyphSize scCachedStyle::GetKernValue( UCS2 ch1, UCS2 ch2 )
{
    RLU             kern;

    if ( GetDeviceValues() )
        return FIgetDEVKern( fSpec, ch1, ch2 );
    else {
        kern = FIgetRLUKern( fSpec, ch1, ch2 );

        if ( kern != 0 ) {
            if ( fFlowDir.IsHorizontal() )
                return scRoundGS( fSetConv * kern );
            return scRoundGS( fPtConv * kern );
        }
    }
    return 0;
}
```

```
        SetSpec( ts );
        fTimeStamp = ++scCachedStyle::fCacheTime;

        fPtConv     = (REAL)GetGlyphHeight() / scBaseRLUsystem;
        fSetConv    = (REAL)GetGlyphWidth()  / scBaseRLUsystem;

        InitWidths();
        ComputeExtentsnCursor();
    }
    else {
        TypeSpec nullSpec;
        SetSpec( nullSpec );
        fPtConv     = 0;
        fSetConv    = 0;
        InitWidths();
        fTimeStamp = 0;
    }

}

/* ================================================================== */

void scCachedStyle::InitFlowdir( const scFlowDir& fd )
{
    fFlowDir = fd;
    InitWidths();
    ComputeExtentsnCursor();
}

/* ================================================================== */

GlyphSize scCachedStyle::GetEscapement( UCS2 ch )
{
    if ( GetDeviceValues() ) {

        if ( ch >= 256 )
            return FIgetDEVEscapement( fSpec, ch );

        if ( fWidths[ch] == kInvalMP ) {
            if ( GetSmallCaps() && CTIsLowerCase( ch ) )
                fWidths[ch] = scRoundMP( FIgetDEVEscapement( fSpec, ::CTToUpper( ch ) ) * kSmallCapC
orrection );
            else
                fWidths[ch] = FIgetDEVEscapement( fSpec, GetCorrectedGlyph( ch ) );
        }
    }
    else {
        REAL        conversion;

        if ( fFlowDir.IsHorizontal() )
            conversion = fSetConv;
        else
            conversion = fPtConv;

        if ( ch >= 256 ) {
            RLU rluWidth = FIgetRLUEscapement( fSpec, ch );
            if ( rluWidth == scBaseRLUsystem ) {
                if ( fFlowDir.IsHorizontal() )
                    return GetSetSize();
                else
                    return GetPtSize();
            }
            return scRoundGS( conversion * rluWidth ) + GetOptLSP();
        }

        if ( fWidths[ch] == kInvalMP ) {
                // it has not been previously computed
            switch ( ch ) {
                default:
                    {
                        RLU rluWidth;
                        if ( GetSmallCaps() && CTIsLowerCase( ch ) )
                            rluWidth = (RLU)scRoundGS( FIgetRLUEscapement( fSpec, ::CTToUpper( ch )
            fInkExtents.x2  = rect.x2;
        }
        else {
            fInkExtents.y1  = 0;
            fInkExtents.y2  = rect.y1 - rect.y2;
            fInkExtents.x1  = -(rect.x2/2) - rect.x1;
            fInkExtents.x2  = rect.x2/2;
        }
    }
    else {
        scRLURect       rect;
        RLU             a,b,c,d;

        FIgetRLUFontExtents( fSpec, a, b, c, d, rect );

        if ( fFlowDir.IsHorizontal() ) {
            scAssert( rect.Valid( eFirstQuad ) );
            rect.FirstToFourth( scBaseRLUsystem );
        }
        scAssert( rect.Valid( eFourthQuad ) );

        fInkExtents.y1  = scRoundMP( fPtConv * rect.rluTop );
        fInkExtents.y2  = scRoundMP( fPtConv * rect.rluBottom );
        fInkExtents.x1  = scRoundMP( fSetConv * rect.rluLeft );
        fInkExtents.x2  = scRoundMP( fSetConv * rect.rluRight );

    }

    if (  GetHorzOblique()  )
        obliqOff = (REAL)tan( AngleToRadians( GetHorzOblique() ) );

    if ( GetHorzOblique() < 0 )
         fInkExtents.x1 += scRoundMP( GetPtSize() * obliqOff );

    if ( GetHorzOblique() < 0 )
        fInkExtents.x2 += scRoundMP( GetPtSize() * obliqOff );

    scAssert( fInkExtents.Valid( eFourthQuad ) );

    if ( fFlowDir.IsHorizontal() )
        fInkExtents.Translate( 0, -GetBaseline() );
    else
        fInkExtents.Translate( GetBaseline(), 0 );

    if ( fFlowDir.IsHorizontal() ) {
        scLEADRefData   ld;

        ld.Set( GetPtSize(), fFlowDir );
        fLogicalExtents.Set( 0, -ld.GetAboveLead(),
                             GetSetSize(), ld.GetBelowLead() );

    }
    else
        fLogicalExtents.Set( -GetSetSize()/2, 0, GetSetSize()/2, GetPtSize() );


    if ( fFlowDir.IsHorizontal() ) {
        fCursorY1       = -scRoundMP( fPtConv * RLU_BASEfmTop );
        fCursorY2       = scRoundMP( fPtConv * RLU_BASEfmBottom );
    }
    else {
        fCursorX1       = -GetSetSize() / 2;
        fCursorX2       = GetSetSize() / 2;
    }
}

/* ================================================================== */

void scCachedStyle::Init( TypeSpec& ts )
{
    if ( ts.ptr() ) {
        TSGetStyle( ts, *this );
```

```
}

/* ==================================================================== */

MicroPoint scCachedStyle::GetParaSpace( scContUnit* cu1,
                                        scContUnit* cu2 )
{
    SetParaStyle( cu1, cu1->GetDefaultSpec() );
    MicroPoint below = cachedParaStyle_.GetSpaceBelow();

    SetParaStyle( cu2, cu2->GetDefaultSpec() );
    MicroPoint above = cachedParaStyle_.GetSpaceAbove();

    return below + above;
}

/* ==================================================================== */

MicroPoint scCachedStyle::GetMaxParaSpace( scContUnit* cu1,
                                           scContUnit* cu2 )
{
    SetParaStyle( cu1, cu1->GetDefaultSpec() );
    MicroPoint below = cachedParaStyle_.GetMaxSpaceBelow();

    SetParaStyle( cu2, cu2->GetDefaultSpec() );
    MicroPoint above = cachedParaStyle_.GetMaxSpaceAbove();

    return below + above;
}

/* ==================================================================== */

void scCachedStyle::SetFlowdir( const scFlowDir& fd )
{
    if ( scCachedStyle::fFlowDir != fd ) {
        int i;

        for ( i = 0; i < fEntries; i++ ) {
            if ( fCachedStyles[i].GetSpec().ptr() )
                fCachedStyles[i].InitFlowdir( fd );
        }
    }
}

/* ==================================================================== */

inline void scCachedStyle::InitWidths( )
{
    register i;
    for ( i = 0; i < 256; i++ )
        fWidths[i] = kInvalMP;
}

/* ==================================================================== */

void scCachedStyle::ComputeExtentsnCursor( void )
{
    REAL            obliqOff;

    if ( GetDeviceValues() ) {
        scXRect         rect;
        MicroPoint      a,b,c,d;

        FIgetDEVFontExtents( fSpec, a, b, c, d, rect );

        if ( fFlowDir.IsHorizontal() ) {
            scAssert( rect.Valid( eFirstQuad ) );
            rect.FirstToFourth( GetPtSize() );
            scAssert( rect.Valid( eFourthQuad ) );

            fInkExtents.y1  = rect.y1;
            fInkExtents.y2  = rect.y2;
            fInkExtents.x1  = rect.x1;
            oldest = i;
        }
    }
    scAssert( oldest >= 0 );

    return oldest;
}

/* ================================================================== */

scCachedStyle& scCachedStyle::GetCachedStyle( TypeSpec& ts )
{
    if ( fLast >= 0 ) {
        if ( fCachedStyles[fLast].GetSpec().ptr() == ts.ptr() )
            return fCachedStyles[fLast];
    }

    return FindCachedStyle( ts );
}


/* ================================================================== */

scCachedStyle&  scCachedStyle::FindCachedStyle( TypeSpec& ts )
{
    int i;

    scAssert( ts.ptr() );

    for ( i = 0; i < fEntries; i++ ) {
        if ( fCachedStyles[i].GetSpec().ptr() == ts.ptr() ) {
            fLast = i;
            SCDebugTrace( 0, "scCachedStyle::FindCachedStyle: found %d\n", fLast );

            return fCachedStyles[i];
        }
    }

    int oldest = GetOldestIndex();

    fCachedStyles[oldest].Init( ts );
    fLast = oldest;

    SCDebugTrace( 0, "scCachedStyle::FindCachedStyle: new cache %d\n", oldest );

    return fCachedStyles[oldest];           // now the newest


/* ================================================================== */

void scCachedStyle::StyleInvalidateCache( TypeSpec& ts )
{
    int i;

    TypeSpec nullSpec;

    for ( i = 0; i < fEntries; i++ ) {
        if ( ts.ptr() == fCachedStyles[i].GetSpec().ptr() )
            fCachedStyles[i].Init( nullSpec );
        else
            fCachedStyles[i].Init( nullSpec );
    }
    cachedParaStyle_.Init( nullSpec );
}

/* ================================================================== */

void scCachedStyle::SetParaStyle( const scContUnit* cu, TypeSpec& ts )
{
    if ( ts.ptr() == cachedParaStyle_.fSpec.ptr() )
        return;
    cachedPara_ = cu;
    cachedParaStyle_.Init( ts );
```

```
/***************************************************************************

    File:       SCSTCACH.CPP


    $Header: /Projects/Toolbox/ct/SCSTCACH.CPP 4     6/17/97 4:16p Wmanis $

    Contains:   Code for the style cache sub-system within the
                Stonehand Composition Toolbox.

    Written by: Manis


    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.


***************************************************************************/

#include "scstcach.h"
#include "sccallbk.h"
#include "sccharex.h"
#include "screfdat.h"
#include "scparagr.h"
#include "scctype.h"
#include <math.h>

scCachedStyle*  scCachedStyle::fCachedStyles;
int             scCachedStyle::fEntries;

int             scCachedStyle::fLast;
long            scCachedStyle::fCacheTime;
scFlowDir       scCachedStyle::fFlowDir( eRomanFlow );

scCachedStyle       scCachedStyle::cachedParaStyle_;
const scContUnit*   scCachedStyle::cachedPara_;

/* ================================================================== */

void scCachedStyle::BuildCache( int entries )
{
    fCachedStyles   = new scCachedStyle [entries];
    fEntries        = entries;
    fLast           = -1;
    fLast           = -1;
    cachedPara_     = 0;
}

/* ================================================================== */

void scCachedStyle::DeleteCache( void )
{
    delete [] fCachedStyles, fCachedStyles = 0;
}

/* ================================================================== */

int scCachedStyle::GetOldestIndex( void )
{
    int     i,
            oldest = -1;
    long    oldestTime = LONG_MAX;

    for ( i = 0; i < fEntries; i++ ) {
        if ( oldestTime > fCachedStyles[i].fTimeStamp ) {
            oldestTime = fCachedStyles[i].fTimeStamp;
```

00620370.073400

```
    MicroPoint      GetRunAroundBorder( void ) const    { return fRunAroundBorder; }
    void            SetRunAroundBorder( MicroPoint b )  { fRunAroundBorder = b; }


    MicroPoint      GetCursorY1( void ) const   { return fCursorY1; }
    MicroPoint      GetCursorX1( void ) const   { return fCursorX1; }
    MicroPoint      GetCursorY2( void ) const   { return fCursorY2; }
    MicroPoint      GetCursorX2( void ) const   { return fCursorX2; }


    void            GetParaBreak( scParaColBreak& pb )
                        {
                            pb.Set( GetLinesBefore(), GetLinesAfter(), GetNoBreak(), GetKeepWithNext
() );
                        }

    static  MicroPoint GetParaSpace( scContUnit*, scContUnit* );
    static  MicroPoint GetMaxParaSpace( scContUnit*, scContUnit* );

    MicroPoint      HorzCompute( RLU rlu )  { return scRoundMP( fSetConv * rlu ); }
    MicroPoint      VertCompute( RLU rlu )  { return scRoundMP( fPtConv * rlu ); }

    eFntBaseline    GetOperativeBaseline( const scFlowDir& fd )
                        {
                            return fd.IsVertical() ? GetVertBaseline() : GetHorzBaseline();
                        }
private:
    void            Init( TypeSpec& ts );
    void            InitFlowdir( const scFlowDir& fd );
    void            InitWidths( void );

    void            ComputeExtentsnCursor( void );

    long            fTimeStamp;

    TypeSpec        fSpec;


    MicroPoint      fRunAroundBorder;

    scXRect         fInkExtents;
    scXRect         fLogicalExtents;

    REAL            fPtConv;
    REAL            fSetConv;

    MicroPoint      fWidths[256];

    union {
        MicroPoint  fCursorY1;
        MicroPoint  fCursorX1;
    };
    union {
        MicroPoint  fCursorY2;
        MicroPoint  fCursorX2;
    };
};

#define gfmS    scCachedStyle::GetCurrentCache( )

inline scCachedStyle& cachedTS( TypeSpec& ts )
{
    return scCachedStyle::GetCachedStyle( ts );
}

inline scCachedStyle& currentTS( )
{
    return scCachedStyle::GetCurrentCache( );
}

#endif
```

```
                                    // set and get the flowdir for the cached values
                                    //
    static void             SetFlowdir( const scFlowDir& fd );
    const scFlowDir&        GetFlowdir( void ) const
                                    {
                                        return fFlowDir;
                                    }

private:
                            // the cached values
    static scCachedStyle*   fCachedStyles;
    static int              fEntries;           // number of entries in the cache
    static int              fLast;


    static scCachedStyle        cachedParaStyle_;
    static const scContUnit*    cachedPara_;

                            // a timer that increments only on loading a cache
    static long             fCacheTime;


                            // for now we are cacheing only on a flow basis
    static scFlowDir        fFlowDir;

                            // get the oldest cached value
    static int              GetOldestIndex( void );

public:
                        scCachedStyle() :
                            fTimeStamp( 0 ),
                            fRunAroundBorder( 0 ),
                            fPtConv( 0 ),
                            fSetConv( 0 )
                                {
                                    SCmemset( fWidths, 0, sizeof( fWidths ) );
                                    fCursorY1 = 0, fCursorY2 = 0;
                                }

                        ~scCachedStyle()
                                {
                                }


    void                SetSpec( TypeSpec& ts )
                            {
                                fSpec = ts;
                            }
    TypeSpec&           GetSpec( void )
                            {
                                return fSpec;
                            }

                            // return the nominal escapement for this glpyh
    GlyphSize           GetEscapement( UCS2 ch );

    GlyphSize           GetLeftHangValue( UCS2 ch )
                            { return scRoundGS( (REAL)GetEscapement( ch ) * GetLeftHang() / 10000 ); }

    GlyphSize           GetRightHangValue( UCS2 ch )
                            { return scRoundGS( (REAL)GetEscapement( ch ) * GetRightHang() / 10000 );
 }

    GlyphSize           GetKernValue( UCS2, UCS2 );

    const scXRect&      GetLogicalExtents( void ) const { return fLogicalExtents; }

    const scXRect&      GetInkExtents( void ) const     { return fInkExtents; }

    eFntBaseline        GetBaselineType( void ) const
                            { if ( fFlowDir.IsVertical() )  return GetVertBaseline();
                              else return GetHorzBaseline(); }
```

```
/***************************************************************************

    File:       SCSTCACH.H

    $Header: /Projects/Toolbox/ct/SCSTCACH.H 3     5/30/97 8:45a Wmanis $

    Contains:   Style cache code.

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

***************************************************************************/

#ifndef _H_SCSTCACH
#define _H_SCSTCACH

#include "scstyle.h"
#include "scmem.h"


inline MicroPoint SCRLUCompMP( MicroPoint size, RLU rlu )
    {
        return scRoundMP( (REAL)size * rlu / scBaseRLUsystem );
    }

inline GlyphSize SCRLUCompGS( GlyphSize size, RLU rlu )
    {
        return scRoundGS( (REAL)size * rlu / scBaseRLUsystem );
    }


class scCachedStyle : public scStyle {

public:
    static void             BuildCache( int entries );
    static void             DeleteCache( void );

    static scCachedStyle&   GetParaStyle( )
                                {
                                    scAssert( cachedPara_ );
                                    return cachedParaStyle_;
                                }
    static void             SetParaStyle( const scContUnit*, TypeSpec& ts );

                                // given a spec, get its cached value,
                                // loading the cache in necessary
                                //
    static scCachedStyle&   GetCachedStyle( TypeSpec& ts );

    static scCachedStyle&   FindCachedStyle( TypeSpec& ts );


                                // this simply returns the last cache we got
                                // hold of, TEMPORARY
    static scCachedStyle&   GetCurrentCache( )
                                {
                                    return fCachedStyles[fLast];
                                }

                                // invalidate the spec, if NULL all specs
                                // will be invalidated
                                //
    static void             StyleInvalidateCache( TypeSpec& ts );
```

```
                                                select_.fMark.fOffset,
                                                select_.fPoint.fOffset );

    diff = select_.fPoint.fOffset - diff;

    if ( select_.fPoint.fPara == range_.fPoint.fPara )
        range_.fPoint.fOffset += diff;      // extend the parent selection

    return ret;
}

/* ==================================================================== */

void stFindIterImp::range( scStreamLocation& mark, scStreamLocation& point )
{
    range_.Decompose( mark, point );
}

/* ==================================================================== */

status SCSTR_GetFindIter( scStream*             str,
                          stUnivString&         ustr,
                          const SearchState&        flags,
                          stFindIter*&  friter )
{
    status stat = scSuccess;

    try {
        friter = new stFindIterImp( ustr, flags, str );
    }
    IGNORE_RERAISE;

    return stat;
}

/* ==================================================================== */

status SCSEL_GetFindIter( scSelection*          sel,
                          stUnivString&         ustr,
                          const SearchState&        flags,
                          stFindIter*&  friter )
{
    status stat = scSuccess;

    try {
        friter = new stFindIterImp( ustr, flags, sel );
    }
    IGNORE_RERAISE;

    return stat;
}

/* ==================================================================== */
```

```
    scContUnit* firstcu = select_.GetMark().fPara;
    scContUnit* lastcu  = range_.GetPoint().fPara;

    int32   startoffset;
    int32   endoffset;

    while ( firstcu != lastcu->GetNext() ) {
        if ( firstcu == select_.GetPoint().fPara )
            startoffset = select_.GetPoint().fOffset;
        else
            startoffset = 0;

        if ( firstcu == range_.GetPoint().fPara )
            endoffset   = range_.GetPoint().fOffset;
        else
            endoffset = LONG_MAX;

        int32   offset;
        if ( firstcu->FindString( ustr_,
                                  state_,
                                  startoffset,
                                  endoffset,
                                  offset ) ) {
            select_.SetMark( TextMarker( firstcu, firstcu->GetCount(), offset ) );
            select_.SetPoint( TextMarker( firstcu, firstcu->GetCount(), offset + ustr_.len ) );
            return 1;
        }
        firstcu = firstcu->GetNext();
    }
    return 0;
}

/* ================================================================== */

int stFindIterImp::backwards()
{
    scContUnit* firstcu = range_.GetMark().fPara;
    scContUnit* lastcu  = select_.GetPoint().fPara;

    int32   startoffset;
    int32   endoffset;

    while ( firstcu != lastcu->GetPrev() ) {
        if ( firstcu == select_.GetMark().fPara )
            endoffset   = select_.GetMark().fOffset;
        else
            endoffset = LONG_MAX;

        if ( firstcu == range_.GetMark().fPara )
            startoffset = range_.GetMark().fOffset;
        else
            startoffset = 0;

        int32   offset;
        if ( firstcu->FindString( ustr_,
                                  state_,
                                  startoffset,
                                  endoffset,
                                  offset ) ) {
            select_.SetMark( TextMarker( firstcu, firstcu->GetCount(), offset ) );
            select_.SetPoint( TextMarker( firstcu, firstcu->GetCount(), offset + ustr_.len ) );
            return 1;
        }
    }
    return 0;
}

/* ================================================================== */

int stFindIterImp::replacetoken( stUnivString& ustr )
{
    int diff = select_.fPoint.fOffset;
    int ret = select_.fMark.fPara->ReplaceToken( ustr,
    range_.SetPoint( TextMarker( point, point->GetCount(), point->GetContentSize() ) );
    select_.SetMark( range_.GetMark() );
    select_.SetPoint( range_.GetMark() );

    select_.Sort();
    range_.Sort();

    reset();
}

/* ================================================================== */

stFindIterImp::stFindIterImp( stUnivString& ustr,
                              const SearchState& state,
                              scSelection*  sel ) :
                                ustr_( ustr ),
                                str_( sel->GetStream() ),
                                range_( *sel ),
                                state_( state ),
                                cuOffset_( 0 )
{
    select_.SetMark( range_.GetMark() );
    select_.SetPoint( range_.GetMark() );

    select_.Sort();
    range_.Sort();

    reset();
}

/* ================================================================== */

void stFindIterImp::release()
{
    delete this;
}

/* ================================================================== */

void stFindIterImp::reset()
{
    if ( !state_.reverse() ) {
        select_.SetMark( range_.GetMark() );
        select_.SetPoint( range_.GetMark() );
    }
    else {
        select_.SetMark( range_.GetPoint() );
        select_.SetPoint( range_.GetPoint() );
    }
}

/* ================================================================== */

int stFindIterImp::setselection( scSelection* sel )
{
    *sel = select_;
    return 1;
}

/* ================================================================== */

int stFindIterImp::next()
{
    if ( !state_.reverse() )
        return forwards();
    else
        return backwards();
}

/* ================================================================== */

int stFindIterImp::forwards()
{
    status stat = scSuccess;

    try {
        iter = new stContUnitIterImp( str, str->First() );
    }
    IGNORE_RERAISE;

    return stat;
}

/* ================================================================== */

status SCSEL_GetContUnitIter( scSelection* sel, stContUnitIter*& iter )
{
    status stat = scSuccess;

    try {
        iter = new stContUnitIterImp( sel );
    }
    IGNORE_RERAISE;

    return stat;
}

/* ================================================================== */

class stFindIterImp : public stFindIter {
public:
    stFindIterImp();
    stFindIterImp( stUnivString& ustr, const SearchState&, scStream* str );
    stFindIterImp( stUnivString& ustr, const SearchState&, scSelection* sel );

    virtual void    release();
    virtual void    reset();
    virtual int     setselection( scSelection* );
    virtual int     next();
    virtual int     replacetoken( stUnivString& );
    virtual void    range( scStreamLocation&, scStreamLocation& );

    int     forwards();
    int     backwards();

private:
    UniversalString ustr_;
    scStream*       str_;
    scSelection     range_;
    scSelection     select_;
    SearchState     state_;
    int32           cuOffset_;
};


/* ================================================================== */

stFindIterImp::stFindIterImp() :
    cuOffset_( 0 )
{
}

/* ================================================================== */

stFindIterImp::stFindIterImp( stUnivString& ustr,
                              const SearchState& state,
                              scStream* str ) :
                                    ustr_( ustr ),
                                    str_( str ),
                                    state_( state ),
                                    cuOffset_( 0 )
{
    scContUnit* mark    = str->First();
    scContUnit* point   = str->Last();

    range_.SetMark( TextMarker( mark, mark->GetCount(), 0 ) );
```

```
stContUnitIterImp::stContUnitIterImp( scSelection* sel ) :
    stream_( sel->GetStream() ),
    cu_( 0 )
{
    range_ = *sel;
    range_.Sort();
    reset();
}

/* ==================================================================== */

void stContUnitIterImp::release()
{
    delete this;
}

/* ==================================================================== */

void stContUnitIterImp::reset()
{
    cu_ = range_.fMark.fPara;
}

/* ==================================================================== */

int stContUnitIterImp::gettokeniter( stTokenIter*& tokenIter )
{
    if ( cu_ ) {
        int32 start;
        int32 end;

        if ( cu_ == range_.fMark.fPara )
            start = range_.fMark.fOffset;
        else
            start = 0;

        if ( cu_ == range_.fPoint.fPara )
            end  = range_.fPoint.fOffset;
        else
            end = LONG_MAX;
        tokenIter = new stTokenIterImp( cu_, start, end, range_ );
    }
    else
        tokenIter = 0;
    return cu_ != 0;

/* ==================================================================== */

int stContUnitIterImp::next()
{
    cu_ = cu_->GetNext();

    if ( range_.fPoint.fPara ) {
        if ( cu_ == range_.fPoint.fPara->GetNext() )
            cu_ = 0;
    }
    return cu_ ? cu_->GetCount() : -1;
}

/* ==================================================================== */

void stContUnitIterImp::range( scStreamLocation& mark, scStreamLocation& point )
{
    range_.Decompose( mark, point );
}

/* ==================================================================== */

status SCSTR_GetContUnitIter( scStream*          str,
                              stContUnitIter*&   iter )
{
```

```
/* ==================================================================== */

int stTokenIterImp::paraselection( scSelection* sel )
{
    if ( cu_ )
        sel->SetParaSelection( cu_, 0, cu_->GetContentSize() );
    return cu_ != 0;
}

/* ==================================================================== */

int stTokenIterImp::setselection( scSelection* sel )
{
    *sel = select_;
    return 1;
}

/* ==================================================================== */

int stTokenIterImp::next()
{
    if ( select_.fMark.SelectStartSpellWord( true ) ) {
        if ( select_.fMark > select_.fPoint )
            select_.fPoint = select_.fMark;
        if ( select_.NextSpellWord( scSelection::inContUnit ) )
            return select_.fPoint.fOffset <= end_;
    }
    return 0;
}

/* ==================================================================== */

int stTokenIterImp::gettoken( stUnivString& ustr )
{
    if ( !select_.IsSliverCursor() ) {
        ulong tokenSize = select_.ContentSize();
        if ( ustr.len < tokenSize )
            return -select_.ContentSize();

        return cu_->GetToken( ustr, select_.fMark.fOffset, select_.fPoint.fOffset );
    }
    return 0;
}

/* ==================================================================== */

int stTokenIterImp::replacetoken( stUnivString& ustr )
{
    int diff = select_.fPoint.fOffset;
    int ret = cu_->ReplaceToken( ustr,
                                 select_.fMark.fOffset,
                                 select_.fPoint.fOffset );

    diff = select_.fPoint.fOffset - diff;
    if ( ret && diff && end_ < LONG_MAX )
        end_ += diff;

    if ( cu_ == range_.fPoint.fPara )
        range_.fPoint.fOffset = end_;       // extend the parent selection

    return ret;
}

/* ==================================================================== */

stContUnitIterImp::stContUnitIterImp( scStream* str, scContUnit* cu ) :
    stream_( str ),
    cu_( cu )
{
    range_.AllSelect();
}

/* ==================================================================== */
```

```
#include "scappint.h"
#include "scpubobj.h"
#include "scstream.h"
#include "scselect.h"
#include "univstr.h"

class stContUnitIterImp : public stContUnitIter {
public:
    stContUnitIterImp( scStream* str, scContUnit* cu );
    stContUnitIterImp( scSelection* sel );
    virtual void    release();
    virtual void    reset();
    virtual int     gettokeniter( stTokenIter*& );
    virtual int     next();
    virtual void    range( scStreamLocation&, scStreamLocation& );

private:
    scStream*   stream_;
    scContUnit* cu_;
    scSelection range_;
};


class stTokenIterImp : public stTokenIter {
public:
    stTokenIterImp( scContUnit*, int32, int32, scSelection& );

    virtual void    release();
    virtual void    reset();
    virtual int     paraselection( scSelection* );
    virtual int     setselection( scSelection* );
    virtual int     gettoken( stUnivString& );
    virtual int     replacetoken( stUnivString& );
    virtual int     next();

private:
    scSelection& range_;
    scContUnit* cu_;
    scSelection select_;

    int32       start_;
    int32       end_;
};

/* ==================================================================== */

stTokenIterImp::stTokenIterImp( scContUnit* cu, int32 start, int32 end, scSelection& range ) :
    range_( range ),
    cu_( cu ),
    start_( start ),
    end_( end )
{
    reset();
}

/* ==================================================================== */

void stTokenIterImp::release()
{
    delete this;
}

/* ==================================================================== */

void stTokenIterImp::reset()
{
    select_.SetParaSelection( cu_, start_, start_ );
    select_.fMark.SelectStartSpellWord( true );
    select_.fPoint = select_.fMark;
    select_.NextSpellWord();
}
```

```
        minHeight   = MIN( minHeight, verts->y );
    }

    return minHeight < 0 ? maxDepth - minHeight : maxDepth;
}

/* ==================================================================== */

long POLYExternalSize( scVertHandle vertH,
                       long         size )
{
    return size * sizeof( scVertex );
}

/* ==================================================================== */

void POLYtoFile( APPCtxPtr  ctxPtr,
                 IOFuncPtr    writeFunc,
                 scVertHandle vertH,
                 ushort       size )
{
    scVertex*   verts;
    long        xsize = POLYExternalSize( vertH, size );
    scAutoUnlock    h( vertH );

    verts = (scVertex*)*h;

//  SCPIO_WritePolygon( ctxPtr, writeFunc, verts, (size_t)size );
}

/* ==================================================================== */

scVertHandle POLYfromFile( APPCtxPtr    ctxPtr,
                           IOFuncPtr    readFunc,
                           ushort       size )
{
    scVertHandle volatile   vertH = 0;
    scVertex*               verts;

    vertH = (scVertHandle)MEMAllocHnd( (size_t)size * sizeof( scVertex ) );

    try {
        scAutoUnlock    h( vertH );
        verts = (scVertex*)*h;

//      SCPIO_ReadPolygon( ctxPtr, readFunc, verts, (size_t)size );
    } catch ( ... ) {
        MEMFreeHnd( vertH );
        throw;
    }

    return vertH;
}

/* ==================================================================== */
```

```
/*****************************************************************************

    File:       SCPOLYGO.C


    $Header: /Projects/Toolbox/ct/SCPOLYGO.CPP 2     5/30/97 8:45a Wmanis $

    Contains:   Polygon code.

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

*****************************************************************************/

#include "scpolygo.h"
#include "scmem.h"
#include "scfileio.h"

/* ================================================================== */

void POLYDuplicate( scVertHandle*   dstOutHP,
                    ushort&         dstNumVerts,
                    scVertHandle    srcOutH,
                    ushort          srcNumVerts )
{
    scVertex*   srcVert;
    scVertex*   dstVert;

    *dstOutHP = (scVertHandle)MEMAllocHnd( (size_t)srcNumVerts * sizeof(scVertex) );

    scAutoUnlock    h1( *dstOutHP );
    scAutoUnlock    h2( srcOutH );

    dstVert = (scVertex*)*h1;
    srcVert = (scVertex*)*h2;

    SCmemcpy( dstVert, srcVert, srcNumVerts * sizeof( scVertex ) );

    dstNumVerts = srcNumVerts;
}

/* ================================================================== */

ushort POLYCountVerts( const scVertex* verts )
{
    ushort numVerts;

    for ( numVerts = 1; verts->fPointType != eFinalPoint; verts++, numVerts++ )
        ;
    return numVerts;
}

/* ================================================================== */

MicroPoint POLYMaxDepth( scVertHandle vertH )
{
    scVertex*   verts;
    MicroPoint  maxDepth    = LONG_MIN,
                minHeight   = LONG_MAX;
    scAutoUnlock    h( vertH );

    verts = (scVertex*)*h;
    for ( ; verts->fPointType != eFinalPoint; verts++ ) {
        maxDepth    = MAX( maxDepth, verts->y );
```

```
                                long );

#endif

#if defined( MEM_DEBUG )

void         memDumpMetrics( void );

void*        MEMAllocPtrDebug( ulong sz, const char *filename, int line );
scMemHandle MEMAllocHndDebug( ulong sz, const char *filename, int line );

void*        MEMDupPtrDebug( const void *, const char *filename, int line );
scMemHandle MEMDupHndDebug( scMemHandle, const char *filename, int line );

void*        MEMResizePtrDebug( void **, ulong sz, const char* file, int line );
scMemHandle MEMResizeHndDebug( scMemHandle, ulong sz, const char* file, int line );

#define MEMAllocPtr( sz )        MEMAllocPtrDebug( (sz), __FILE__, __LINE__ )
#define MEMAllocHnd( sz )        MEMAllocHndDebug( (sz), __FILE__, __LINE__ )

#define MEMResizeHnd( h, sz )   MEMResizeHndDebug( (h), (sz), __FILE__, __LINE__ )
#define MEMResizePtr( p, sz )   MEMResizePtrDebug( (p), (sz), __FILE__, __LINE__ )

#define MEMDupPtr( p )           MEMDupPtrDebug( (p), __FILE__, __LINE__ )
#define MEMDupHnd( p )           MEMDupHndDebug( (p), __FILE__, __LINE__ )


//void*      MEMAllocObjDebug( ulong sz, const char *filename, int line );
//void*      MEMDupObjDebug( void *, const char *filename, int line );
//#define    MEMAllocObj( sz )   MEMAllocObjDebug( (sz), __FILE__, __LINE__ )
//#define MEMDupObj( p )         MEMDupObjDebug( (p), __FILE__, __LINE__ )

#endif  /* SCDEBUG */

/* ================================================================ */
/* ================================================================ */

class scAutoUnlock {
public:
            scAutoUnlock( scMemHandle hnd );
           ~scAutoUnlock( void );

    void    *operator *() { return scMemDeref( fHandle ); }

private:
    scMemHandle fHandle;
};

/* ================================================================ */

// The scStackMem is a convenient way to allocate some temporary
// memopry without having to worry about freeing it.
// since it is rather unsafe to create a stack object that
// allocates memory we will create the next best thing,
// the object will be created storing the desired size,
// and then the fucntion can get the memory by calling Init.
// The memory will be freed by the contstructor or may be freed
// by the user using Free. Resize can resize the memory if needed

/* ================================================================ */
/* ================================================================ */

#endif /* _H_MEM */
```

```
scMemHandle MEMDupHnd(
                scMemHandle hnd );      // @parm Handle to dup.

//void*     MEMDupObj( void * );

#endif


// @CALLBACK Free a pointer.
void        MEMFreePtr(
                void *ptr );            // @parm Pointer to free.

// @CALLBACK Free a handle.
void        MEMFreeHnd(
                scMemHandle hnd );      // @parm Handle to free.
//void      MEMFreeObj( void * );


// @CALLBACK Resize a pointer.
void*       MEMResizePtr(
                void**  ptr,        // @parm Pointer to resize.
                ulong       sz );   // @parm New size in bytes.


// @CALLBACK Resize a Handle.
scMemHandle MEMResizeHnd(
                scMemHandle hnd,    // @parm Handle to resize.
                ulong       sz );   // @parm New size in bytes.

// @CALLBACK Get size in bytes of pointer.
ulong       MEMGetSizePtr(
                const void* ptr );      // @parm Pointer to size.

// @CALLBACK Get size in bytes of handle.
ulong       MEMGetSizeHnd(
                scMemHandle hnd ); // @parm Handle to size.

//ulong     MEMGetSizeObj( void * );

// @CALLBACK Lock a handle, returns ptr to handle contents.
void*       MEMLockHnd(
                scMemHandle     hnd,            // @parm Handle to lock.
                int             counted = 1 );  // @parm If non-zero count the locks.


// @CALLBACK Unlock a handle.
void        MEMUnlockHnd(
                scMemHandle     hnd,            // @parm Handle to unlock.
                int counted = 0 );              // @parm If non-zero count the locks.


#if SCDEBUG < 2
    inline void MEMValidate( void * ){}
#else
    void    MEMValidate( void *ptr );
#endif


#ifndef SCmemset        // we are in a 16 bit world

void scFar * scFar scCDecl SCmemset( void scFar *,
                                     int,
                                     long );

void scFar * scFar scCDecl SCmemmove( void scFar *,
                                      const void scFar *,
                                      long );

void scFar * scFar scCDecl SCmemcpy(    void scFar *,
                                        const void scFar *,
                                        long );
int scFar scCDecl SCmemcmp( const void scFar *,
                            const void scFar *,
/*=================================================================

    File:       MEM.H

    $Header: /Projects/Toolbox/ct/SCMEM.H 2     5/30/97 8:45a Wmanis $

    Contains:   Memory bottle neck fucntions. These functions should be replaced
                by the client - integrating stonehand memory management and
                the clients memory management.

    Written by: Sealy

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

    @doc

==================================================================*/

#ifndef _H_MEM
#define _H_MEM

#ifdef SCMACINTOSH
    #pragma once
#endif

#include "scexcept.h"

#ifndef useSMARTHEAP
    typedef long MEM_POOL;
#endif

// @struct scPoolInfo | A structure for allocating mem pools.
struct scPoolInfo {
    size_t      fBlockSize;     // @field Sizeof block for pool.
    MEM_POOL    fPool;          // @field See Smart Heap, for use by MEMInit.
};


// @CALLBACK This initializes memory for use by Composition Toolbox.
// Called from <f SCENG_Init>.
//
void        MEMInit(
                    scPoolInfo pools[]  );      // @parm <t scPoolInfo> array is null terminated.

// @CALLBACK Called when Toolbox is closed with <f SCENG_Fini>.
void        MEMFini( void );


#if !defined( MEM_DEBUG )

// @CALLBACK Allocate fixed block, return ptr.
void*       MEMAllocPtr(
                ulong sz ); // @parm Size in bytes.

// @CALLBACK Allocate moveable block, return handle that can be dereferenced with **.
scMemHandle MEMAllocHnd(
                ulong sz ); // @parm Size in bytes.

//void*     MEMAllocObj( ulong sz );    // allocate using object factory, return ptr

// @CALLBACK Duplicate a pointer.
void*       MEMDupPtr(
                 const void *ptr );     // @parm Pointer to dup.

// @CALLBACK Duplicate a handle.
```

09623970.072400

```
/* ================================================================== */

static void MEMArrayPtrTest()
{
    short   i,
            element;

    scMemArray *ptrArr;

    ptrArr = SCNEW scMemArray( sizeof(short) );

    for ( i = 0; i < 100; i++ )
        ptrArr->AppendData( (ElementPtr)&i );

    for ( i = 0; i < ptrArr->GetNumItems(); i++ ) {
        ptrArr->GetDataAt( i, (ElementPtr)&element );
        scAssert( i == element );
    }

    ptrArr->RemoveDataAt( 0 );


    for ( i = 0; i < ptrArr->GetNumItems(); i++ ) {
        ptrArr->GetDataAt( i, (ElementPtr)&element );
        scAssert( i+1 == element );
    }

    ptrArr->RemoveAll();

    delete ptrArr;
}

/* ================================================================== */

static void MEMArrayHndTest()
{
    short   i,
            element;

    scHandleArray ptrArr( sizeof(short) );

    for ( i = 0; i < 100; i++ )
        ptrArr.AppendData( (ElementPtr)&i );

    for ( i = 0; i < ptrArr.GetNumItems(); i++ ) {
        ptrArr.GetDataAt( i, (ElementPtr)&element );
        scAssert( i == element );
    }

    ptrArr.RemoveDataAt( 0 );


    for ( i = 0; i < ptrArr.GetNumItems(); i++ ) {
        ptrArr.GetDataAt( i, (ElementPtr)&element );
        scAssert( i+1 == element );
    }

    ptrArr.RemoveAll();
}

/* ================================================================== */

void MEMArrayTest( void )
{
    MEMArrayPtrTest();
    MEMArrayHndTest();
}

/* ================================================================== */

#endif
```

```
}

/* ==================================================================== */

void scHandleArray::SizeSlots( long numItems )
{
        // do not shrink if we are retaining memory or if no resizing is
        // necessary
        //
    if ( ( numItems < fElemSlots && fRetainMem ) || fElemSlots == numItems )
        return;

    long oldSize = fElemSlots;
    raise_if( fItems == NULL, scERRmem );
    fItems      = MEMResizeHnd( (scMemHandle)fItems, fElemSize * numItems );
    fElemSlots  = numItems;
    ClearMem( oldSize );
}

/* ==================================================================== */

ElementPtr scHandleArray::Lock( void ) const
{
    raise_if( fItems == NULL, scERRmem );
    return (ElementPtr)MEMLockHnd( (scMemHandle)fItems );
}

/* ==================================================================== */

void scHandleArray::Unlock( void ) const
{
    raise_if( fItems == NULL, scERRmem );
    MEMUnlockHnd( (scMemHandle)fItems );
}

/* ==================================================================== */

scHandleArray&  scHandleArray::operator=( const scHandleArray& cpa )
{
    if ( fItems )
        MEMResizeHnd( &fItems, cpa.fElemSize * cpa.fElemSlots );
    else
        fItems = MEMAllocHnd( cpa.fElemSize * cpa.fElemSlots );

    scAbstractArray::operator=( cpa );

    scMemHandle     h1_ = (scMemHandle)fItems;
    scMemHandle     h2_ = (scMemHandle)cpa.fItems;

    scAutoUnlock    h1( h1_ );
    scAutoUnlock    h2( h2_ );

    memcpy( *h1, *h2, fElemSize * fNumItems );

    return *this;
}


/* ==================================================================== */
/* ==================================================================== */
/*                        MEMARRAYTEST                                   */
/* ==================================================================== */
/* ==================================================================== */

#define MEMARRAYTEST

#ifdef MEMARRAYTEST

#include <windows.h>
#include "assert.h"
```

```
        //
    if ( ( numItems < fElemSlots && fRetainMem ) || fElemSlots == numItems )
        return;

    long oldSize = fElemSlots;
    raise_if( fItems == NULL, scERRmem );
    MEMResizePtr( &fItems, fElemSize * numItems );
    fElemSlots = numItems;
    ClearMem( oldSize );
}

/*=======================================================================*/

ElementPtr scMemArray::Lock() const
{
    raise_if( fItems == NULL, scERRmem );
    return (ElementPtr)fItems;
}

/* ===================================================================== */

scMemArray& scMemArray::operator=( const scMemArray& cpa )
{
    if ( fItems )
        MEMResizePtr( &fItems, cpa.fElemSize * cpa.fElemSlots );
    else
        fItems = MEMAllocPtr( cpa.fElemSize * cpa.fElemSlots );

    scAbstractArray::operator=( cpa );
    SCmemcpy( fItems, cpa.fItems, fElemSize * fNumItems );

    return *this;
}

/* ===================================================================== */
/* ===================================================================== */

scHandleArray::scHandleArray( size_t    elemSize,
                              unsigned  clearmem ) :
                                scAbstractArray( elemSize, clearmem )
{
    fItems      = (scMemHandle)MEMAllocHnd( fElemSize * fBlockSize );
    fElemSlots  = fBlockSize;
    ClearMem( 0 );
}

/* ===================================================================== */

scHandleArray::~scHandleArray()
{
    if ( fItems )
        MEMFreeHnd( (scMemHandle)fItems ), fItems = 0;
}

/* ===================================================================== */

int scHandleArray::IsEqual( const scObject& obj ) const
{
    const scHandleArray&    harray  = (const scHandleArray&)obj;
    if ( fNumItems != harray.fNumItems )
        return 0;
    return scAbstractArray::IsEqual( obj );
}

/* ===================================================================== */

void scHandleArray::GrowSlots( long newItems )
{
    long oldSize = fElemSlots;
    raise_if( fItems == NULL, scERRmem );
    fItems = MEMResizeHnd( (scMemHandle)fItems, fElemSize * ( fElemSlots + newItems ) );
    fElemSlots += newItems;
    ClearMem( oldSize );
```

```
        // items
    const scAbstractArray&  absarray    = ((const scAbstractArray&)obj);
    ElementPtr              elemPtr     = Lock();
    ElementPtr              elemPtr2    = absarray.Lock();
    int                     isEqual;

    isEqual = ( fElemSize == absarray.fElemSize );
    if ( isEqual )
        isEqual = !SCmemcmp( elemPtr, elemPtr2, fNumItems * fElemSize );

    Unlock();
    absarray.Unlock();

    return isEqual;
}

/* ==================================================================== */

scMemArray::scMemArray( size_t      elemSize,
                        unsigned    clearmem ) :
                            scAbstractArray( elemSize, clearmem )
{
    fItems      = MEMAllocPtr( fElemSize * fBlockSize );
    fElemSlots  = fBlockSize;
    ClearMem( 0 );
}


/* ==================================================================== */

scMemArray::scMemArray( const scMemArray& ma ) :
    scAbstractArray( ma )
{
    fItems      = MEMAllocPtr( fElemSize * ma.fElemSlots );
    fElemSlots  = ma.fElemSlots;
    SCmemcpy( fItems, ma.fItems, fElemSize * ma.fElemSlots );
}

/* ==================================================================== */

scMemArray::~scMemArray()
{
    if ( fItems )
        MEMFreePtr( fItems ), fItems = 0;
}

/* ==================================================================== */

int scMemArray::IsEqual( const scObject& obj ) const
{
    const scHandleArray&    harray  = (const scHandleArray&)obj;
    if ( fNumItems != harray.GetNumItems() )
        return 0;
    return scAbstractArray::IsEqual( obj );
}

/* ==================================================================== */

void scMemArray::GrowSlots( long newItems )
{
    long oldSize = fElemSlots;
    raise_if( fItems == NULL, scERRmem );
    MEMResizePtr( &fItems, fElemSize * ( fElemSlots + newItems ) );
    fElemSlots += newItems;
    ClearMem( oldSize );
}

/* ==================================================================== */

void scMemArray::SizeSlots( long numItems )
{
        // do not shrink if we are retaining memory or if no resizing is
        // necessary
```

```
                                  long              param ) const
{
    ElementPtr  elemPtr = Lock();
    int     i;

    for ( i = 0; i < fNumItems; i++, elemPtr += fElemSize ) {
        if ( (*func)( elemPtr, param ) ) {
            if ( --nth == 0 )
                break;
        }
    }

    Unlock();

    return i < fNumItems ? i : -1;
}

/* ==================================================================== */

long scAbstractArray::NthSuccess( CBoolConstFunc2   func,
                                  long              nth,
                                  long              param1,
                                  long              param2  ) const
{
    int         i;
    ElementPtr  elemPtr = Lock();

    for ( i = 0; i < fNumItems; i++, elemPtr += fElemSize ) {
        if( (*func)( elemPtr, param1, param2 ) ) {
            if ( --nth == 0 )
                break;
        }
    }

    Unlock();

    return i < fNumItems ? i : -1;
}

/* ==================================================================== */

void scAbstractArray::QuickSort( CPtrArrayCmpFunc compfunc )
{
    raise_if( fItems == NULL, scERRmem );

    ElementPtr  elemPtr = Lock();

    ::qsort( elemPtr, fNumItems, fElemSize, compfunc );

    Unlock();
}

/*====================================================================*/

scAbstractArray&    scAbstractArray::operator=( const scAbstractArray& absarray )
{
    fNumItems   = absarray.fNumItems;
    fElemSlots  = absarray.fElemSlots;
    fElemSize   = absarray.fElemSize;
    fBlockSize  = absarray.fBlockSize;

    return *this;
}

/* ==================================================================== */
// this is a very dumb and dangerous method if you are using the abstract
// array class to hold structures, structure alignment may cause there to
// be "dead" space in the structure which may be filled with
// gargbage --- rendering this method useless, so overide if needed

int scAbstractArray::IsEqual( const scObject& obj ) const
{
        // by the time we hit here we assume we have the same number of
    Unlock();

    return i < fNumItems ? i : -1;
}

/* ==================================================================== */

long scAbstractArray::NthSuccess( CBoolFunc1    func,
                                  long          nth,
                                  long          param )
{
    long        i;
    ElementPtr  elemPtr = Lock();

    for ( i = 0; i < fNumItems; i++, elemPtr += fElemSize ) {
        if ( (*func)( elemPtr, param ) ) {
            if ( --nth == 0 )
                break;
        }
    }

    Unlock();

    return i < fNumItems ? i : -1;
}

/* ==================================================================== */

long scAbstractArray::NthSuccess( CBoolFunc2    func,
                                  long          nth,
                                  long          param1,
                                  long          param2  )
{
    long        i;
    ElementPtr  elemPtr = Lock();

    for ( i = 0; i < fNumItems; i++, elemPtr += fElemSize ) {
        if( (*func)( elemPtr, param1, param2 ) ) {
            if ( --nth == 0 )
                break;
        }
    }

    Unlock();

    return i < fNumItems ? i : -1;
}

/* ==================================================================== */

long scAbstractArray::NthSuccess( CBoolConstFunc0   func,
                                  long              nth ) const
{
    int         i;
    ElementPtr  elemPtr = Lock();

    for ( i = 0; i < fNumItems; i++, elemPtr += fElemSize ) {
        if ( (*func)( elemPtr ) ) {
            if ( --nth == 0 )
                break;
        }
    }

    Unlock();

    return i < fNumItems ? i : -1;
}

/* ==================================================================== */

long scAbstractArray::NthSuccess( CBoolConstFunc1   func,
                                  long              nth,
    Unlock();

    return i < fNumItems ? i : -1;
}

/* ==================================================================== */

long scAbstractArray::FirstSuccess( CBoolConstFunc0 func ) const
{
    int         i;
    ElementPtr  elemPtr = Lock();


    for ( i = 0; i < fNumItems; i++, elemPtr += fElemSize )
        if ( (*func)( elemPtr ) )
            break;

    Unlock();

    return i < fNumItems ? i : -1;

}

/* ==================================================================== */

long scAbstractArray::FirstSuccess( CBoolConstFunc1 func,
                                    long            param ) const
{
    int         i;
    ElementPtr  elemPtr = Lock();

    for ( i = 0; i < fNumItems; i++, elemPtr += fElemSize )
        if ( (*func)( elemPtr, param ) )
            break;

    Unlock();

    return i < fNumItems ? i : -1;
}

/* ==================================================================== */

long scAbstractArray::FirstSuccess( CBoolConstFunc2 func,
                                    long            param1,
                                    long            param2  ) const
{
    int         i;
    ElementPtr  elemPtr = Lock();

    for ( i = 0; i < fNumItems; i++, elemPtr += fElemSize )
        if( (*func)( elemPtr, param1, param2 ) )
            break;

    Unlock();

    return i < fNumItems ? i : -1;
}

/* ==================================================================== */


long scAbstractArray::NthSuccess( CBoolFunc0    func,
                                  long          nth )
{
    long        i;
    ElementPtr  elemPtr = Lock();

    for ( i = 0; i < fNumItems; i++, elemPtr += fElemSize ) {
        if ( (*func)( elemPtr ) ) {
            if ( --nth == 0 )
                break;
        }
    }
```

```
void scAbstractArray::DoForEach( CVoidFunc1 func, long param )
{
    int         i;
    ElementPtr  elemPtr = Lock();

    for ( i = 0; i < fNumItems; i++, elemPtr += fElemSize )
        (*func)( elemPtr, param );

    Unlock();
}

/* ================================================================== */

void scAbstractArray::DoForEach( CVoidFunc2 func, long param1, long param2  )
{
    int         i;
    ElementPtr  elemPtr = Lock();

    for ( i = 0; i < fNumItems; i++, elemPtr += fElemSize )
        (*func)( elemPtr, param1, param2 );

    Unlock();
}

/* ================================================================== */

long scAbstractArray::FirstSuccess( CBoolFunc0 func )
{
    int         i;
    ElementPtr  elemPtr = Lock();

    for ( i = 0; i < fNumItems; i++, elemPtr += fElemSize )
        if ( (*func)( elemPtr ) )
            break;

    Unlock();

    return i < fNumItems ? i : -1;
}

/* ================================================================== */

long scAbstractArray::FirstSuccess( CBoolFunc1  func,
                                    long        param )
{
    int         i;
    ElementPtr  elemPtr = Lock();


    for ( i = 0; i < fNumItems; i++, elemPtr += fElemSize )
        if ( (*func)( elemPtr, param ) )
            break;

    Unlock();

    return i < fNumItems ? i : -1;
}

/* ================================================================== */

long scAbstractArray::FirstSuccess( CBoolFunc2  func,
                                    long        param1,
                                    long        param2 )
{
    int         i;
    ElementPtr  elemPtr = Lock();


    for ( i = 0; i < fNumItems; i++, elemPtr += fElemSize )
        if( (*func)( elemPtr, param1, param2 ) )
            break;
    ElementPtr  ptr = Lock();

    SCmemmove( ptr + ( ( index + elements ) * fElemSize ),
             ptr + ( index * fElemSize ),
             ( fNumItems - index ) * fElemSize  );

    SCmemmove(  ptr + ( index * fElemSize ),
              elemptr,
              elements * fElemSize );

    Unlock();

    fNumItems += elements;
}

/* ==================================================================== */

ElementPtr scAbstractArray::GetDataAt( long         index,
                                       ElementPtr   elemptr,
                                       long         elements ) const
{
    scAssert( index + elements <= fNumItems );

    ElementPtr  ptr = Lock();

    SCmemmove( elemptr,
             ptr + (index * fElemSize),
             elements * fElemSize );

    Unlock();

    return elemptr;
}

/* ==================================================================== */

ElementPtr scAbstractArray::Pop( ElementPtr elemptr,
                                 long       elements )
{
    long index = fNumItems - 1;

    elemptr = GetDataAt( index, elemptr, elements );
    RemoveDataAt( index, elements );
    return elemptr;
}

/* ==================================================================== */

ElementPtr scAbstractArray::GetTop( ElementPtr  elemptr,
                                    long        elements )
{
    long index = fNumItems - 1;

    elemptr = GetDataAt( index, elemptr, elements );
    return elemptr;
}

/* ==================================================================== */

void scAbstractArray::DoForEach( CVoidFunc0 func )
{
    ElementPtr  elemPtr = Lock();
    int     i;

    for ( i = 0; i < fNumItems; i++, elemPtr += fElemSize )
        (*func)( elemPtr );

    Unlock();
}

/* ==================================================================== */
```

```
    }

    catch ( ... ) {
        Unlock();
        throw;
    }

    Unlock();
}

/* ==================================================================== */

void scAbstractArray::Cut( scAbstractArray& arr, long start, long end )
{
    Copy( arr, start, end );
    RemoveDataAt( start, end - start );
}

/* ==================================================================== */

void scAbstractArray::AppendData( const ElementPtr  elemptr,
                                  long              elements )
{
    SetNumSlots( fNumItems + elements );

    ElementPtr  ptr = Lock();
    SCmemmove( ptr + ( (long)fNumItems * fElemSize ),
              elemptr,
              (fElemSize * elements) );
    Unlock();
    fNumItems += elements;

/* ==================================================================== */

void scAbstractArray::RemoveDataAt( long index,
                                    long elements )

    scAssert( index + elements <= fNumItems );

    ElementPtr  ptr = Lock();

    SCmemmove( ptr + ( index * fElemSize ),
              ptr + ( ( index + elements ) * fElemSize ),
              ( fNumItems - index - elements ) * fElemSize  );

    Unlock();

    fNumItems -= elements;

    ShrinkSlots();
}

/* ==================================================================== */

void scAbstractArray::AlterDataAt( long         index,
                                   ElementPtr   elemptr,
                                   long         elements )
{
    ElementPtr  ptr = Lock();
    SCmemmove( ptr + ( (long)index * fElemSize ),
              elemptr,
              fElemSize * elements );
    Unlock();
}

/* ==================================================================== */

void scAbstractArray::InsertAt( long        index,
                                ElementPtr  elemptr,
                                long        elements )
{
    SetNumSlots( fNumItems + elements );
```

```
/*************************************************************************

    File:       scmemarr.cpp

    $Header: /Projects/Toolbox/ct/SCMEMARR.CPP 2     5/30/97 8:45a Wmanis $

    Contains:   Variable sized array code.

    Written by: Manis

    Copyright (c) 1989-1994 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.


*************************************************************************/

#include "scmemarr.h"
#include "scmem.h"

#include <string.h>
#include <stdlib.h>

/* ==================================================================== */

void scAbstractArray::ClearMem( long oldsize )
{
        // either we do need to clear memory or we have shrunk it
    if ( !fClearMem || oldsize >= fElemSlots )
        return;

    ElementPtr  elements = Lock();

    SCmemset( elements + ( oldsize * fElemSize ),
            0,
            ( fElemSlots - oldsize ) * fElemSize );

    Unlock();
}

/* ==================================================================== */

void scAbstractArray::Paste( long index, const scAbstractArray& arr )
{
    ElementPtr ptr = arr.Lock();

    try {
        InsertAt( index, ptr, arr.GetNumItems() );
    }

    catch ( ... ) {
        arr.Unlock();
        throw;
    }

    arr.Unlock();
}

/* ==================================================================== */

void scAbstractArray::Copy( scAbstractArray& arr, long start, long end ) const
{
    ElementPtr ptr = Lock();

    try {
        arr.InsertAt( 0, ptr, end - start );
```

```
                   }
           ~scHandleArrayLock( void)
                   {
                      array_.Unlock();
                   }


    void    *operator *() { return array_.GetMem(); }

private:
    scHandleArray&  array_;
};


#endif /* _H_CMEMARR */
```

```
    ElementPtr      GetMem( void ) const
                    {
                        return (ElementPtr)fItems;
                    }
    ElementPtr      GetMem( long n ) const
                    {
                        return (ElementPtr)((char*)fItems + (n*fElemSize));
                    }

    virtual int     IsEqual( const scObject& ) const;

protected:
    void            GrowSlots( long );
    void            SizeSlots( long );

    ElementPtr      Lock( void ) const;
    void            Unlock( void ) const    {}
};


/* ==================================================================== */
/* ==================================================================== */

class scHandleArrayLock;

class scHandleArray : public scAbstractArray {
    scDECLARE_RTTI;
    friend scHandleArrayLock;
public:
                    scHandleArray() : scAbstractArray( sizeof( void* ) ){}
                    scHandleArray( size_t elemSize, unsigned clearmem = 0 );
                    scHandleArray( const scHandleArray&, unsigned clearmem = 0 );

                    ~scHandleArray();

    scHandleArray&  operator=( const scHandleArray& );

        // NOTE: !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
        // these rely upon the fact the memory manager uses a mac type
        // handle, smart heap does this so if you use smart heap or the
        // stonehand memory manager you are safe
        // NOTE: !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    ElementPtr      GetMem( void ) const
                    {
                        return (ElementPtr)*((char**)fItems);
                    }
    ElementPtr      GetMem( long n ) const
                    {
                        return (ElementPtr)(*((char**)fItems) + (n*fElemSize));
                    }

    virtual int     IsEqual( const scObject& ) const;

protected:
    void            GrowSlots( long );
    void            SizeSlots( long );

    ElementPtr      Lock( void ) const;
    void            Unlock( void ) const;
};


class scHandleArrayLock {
public:
            scHandleArrayLock( scHandleArray* array ) :
                    array_( *array )
                    {
                        array_.Lock();
                    }
            scHandleArrayLock( scHandleArray& array ) :
                    array_( array )
                    {
                        array_.Lock();
    long            FirstSuccess( CBoolConstFunc2, long, long  ) const;

        // these will return the index of the nth
        // successful match of the data
        // NOTE: if their is NO match it RETURNS -1
    long            NthSuccess( CBoolFunc0, long nth );
    long            NthSuccess( CBoolFunc1, long nth, long );
    long            NthSuccess( CBoolFunc2, long nth, long, long  );

    long            NthSuccess( CBoolConstFunc0, long nth ) const;
    long            NthSuccess( CBoolConstFunc1, long nth, long ) const;
    long            NthSuccess( CBoolConstFunc2, long nth, long, long ) const;

    void            QuickSort( CPtrArrayCmpFunc );

    void            SetRetainMem( unsigned tf )
                    {
                        fRetainMem = tf ? 1 : 0;
                    }
    unsigned        GetRetainMem( void ) const
                    {
                        return fRetainMem;
                    }

    virtual void    SetNumSlots( long numSlots )
                    {
                        SizeSlots( ( ( numSlots / fBlockSize ) + 1 ) * fBlockSize );
                    }

protected:
    virtual void    MoreSlots( void )
                    {
                        GrowSlots( fBlockSize );
                    }
    virtual void    SizeSlots( long ) = 0;
    virtual void    GrowSlots( long ) = 0;
    virtual void    ShrinkSlots( void )
                    {
                        SetNumSlots( fNumItems );
                    }
    void            ClearMem( long oldsize );

    virtual ElementPtr  Lock( void ) const = 0;
    virtual void        Unlock( void ) const = 0;

    long        fNumItems;          // num of elements in the array
    long        fElemSlots;         // num of elements potentially in allocated space
    unsigned    fElemSize   : 16;   // element size
    unsigned    fBlockSize  : 8;    // for growing and shrinking we grow in greater
                                    // than one element unit - this is that unit
                                    // typically 4
    unsigned    fClearMem   : 1;    // if this is set this will zero out mem
                                    // that is allocated
    unsigned    fRetainMem  : 1;    // do not shrink memory if this is set
    unsigned    fPad        : 7;
    void*       fItems;             // the data
};


/* ================================================================== */
/* ================================================================== */


class scMemArray : public scAbstractArray {
    scDECLARE_RTTI;
public:
                    scMemArray() : scAbstractArray( sizeof( void* ) ){}
                    scMemArray( size_t elemSize, unsigned clearmem = 0 );
                    scMemArray( const scMemArray& );
                    ~scMemArray();

    scMemArray&     operator=( const scMemArray& );
```

```
                        fElemSlots( 0 ),
                        fBlockSize( aa.fBlockSize ),
                        fClearMem( aa.fClearMem ),
                        fRetainMem( aa.fRetainMem ),
                        fPad( 0 ),
                        fItems( 0 ) {}


                ~scAbstractArray()
                {
                    fElemSize   = 0,
                    fNumItems   = 0,
                    fElemSlots  = 0,
                    fBlockSize  = 4,
                    fItems      = 0;
                }


    scAbstractArray&    operator=( const scAbstractArray& );

    virtual int     IsEqual( const scObject& ) const;

        // use this call extrememly wisely for purposes of copying
    void            SetMemory( void *mem )
                    {
                        fItems = mem;
                    }

    long            GetNumItems( void ) const
                    {
                        return fNumItems;
                    }

    virtual void    RemoveAll( void )
                    {
                        fNumItems=0, ShrinkSlots();
                    }

    virtual void    Paste( long, const scAbstractArray& );
    virtual void    Copy( scAbstractArray&, long, long ) const;
    virtual void    Cut( scAbstractArray&, long, long );

    void            RemoveDataAt( long, long num = 1 );
    void            AppendData( const ElementPtr, long num = 1 );
    void            AlterDataAt( long, ElementPtr, long num = 1 );
    void            InsertAt( long, ElementPtr, long num = 1 );
    ElementPtr      GetDataAt( long, ElementPtr, long num = 1 ) const;

                    // stack operators
    void            Push( const ElementPtr p, long elements = 1 )
                    {
                        AppendData( p, elements );
                    }
    ElementPtr      Pop( ElementPtr, long elements = 1 );

                    // get the top element(s)
    ElementPtr      GetTop( ElementPtr, long elements = 1 );


    void            DoForEach( CVoidFunc0 );
    void            DoForEach( CVoidFunc1, long );
    void            DoForEach( CVoidFunc2, long, long  );


        // these will return the index of the first
        // successful match of the data
        // NOTE: if their is NO match it RETURNS -1
    long            FirstSuccess( CBoolFunc0 );
    long            FirstSuccess( CBoolFunc1, long );
    long            FirstSuccess( CBoolFunc2, long, long  );

    long            FirstSuccess( CBoolConstFunc0 ) const;
    long            FirstSuccess( CBoolConstFunc1, long ) const;
```

```
/*********************************************************************

    File:       scmemarr.h

    $Header: /Projects/Toolbox/ct/SCMEMARR.H 2     5/30/97 8:45a Wmanis $

    Contains:   The variable size array class.

    Written by: Manis

    Copyright (c) 1989-1994 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

*********************************************************************/

#ifndef _H_CMEMARR
#define _H_CMEMARR

#include "sctypes.h"
#include "scobject.h"
#include "scexcept.h"


typedef char scFar *ElementPtr;

typedef void (*CVoidFunc0)( ElementPtr );
typedef void (*CVoidFunc1)( ElementPtr, long );
typedef void (*CVoidFunc2)( ElementPtr, long, long );


typedef Bool (*CBoolFunc0)( ElementPtr );
typedef Bool (*CBoolFunc1)( ElementPtr, long );
typedef Bool (*CBoolFunc2)( ElementPtr, long, long );

typedef Bool (*CBoolConstFunc0)( const ElementPtr );
typedef Bool (*CBoolConstFunc1)( const ElementPtr, long );
typedef Bool (*CBoolConstFunc2)( const ElementPtr, long, long );

extern "C" {
    typedef int (scCDecl* CPtrArrayCmpFunc)(const void *, const void *);
}

/* ==================================================================== */
/* ==================================================================== */
/* ==================================================================== */
/* ==================================================================== */

class scAbstractArray : public scObject {
    scDECLARE_RTTI;
public:
                    scAbstractArray( int elemsize, unsigned clearmem = 0 ) :
                        fElemSize( elemsize ),
                        fNumItems( 0 ),
                        fElemSlots( 0 ),
                        fBlockSize( 4 ),
                        fClearMem( clearmem ),
                        fRetainMem( 0 ),
                        fPad( 0 ),
                        fItems( 0 ) {}


                    scAbstractArray( const scAbstractArray& aa ) :
                            fElemSize( aa.fElemSize ),
                            fNumItems( aa.fNumItems ),
void scFar*  scFar scCDecl SCmemcpy( void scFar*         dst,
                                     const void scFar*   src,
                                     long                len )
{
    return _fmemcpy( dst, src, (size_t)len );
}

/* ==================================================================== */

int scFar scCDecl SCmemcmp( const void scFar*   p1,
                            const void scFar*   p2,
                            long                len )
{
    return _fmemcmp( p1, p2, (size_t)len );
}

#endif

/* ==================================================================== */
```

```
}

/* ======================================================================== */

scMemHandle MEMDupHndDebug( scMemHandle obj, const char *filename, int line )
{
    scMemHandle hnd;

    if ( !RandomFailure() ) {
        ulong   sz = _msize( obj ) - sizeof( MacHandle );

        hnd = MEMAllocHndDebug( sz, filename, line );

        try {
            void*   srcP = MEMLockHnd( obj );
            void*   dstP = MEMLockHnd( hnd );
            SCmemcpy( dstP, srcP, sz );
        }
        catch (...) {
            MEMUnlockHnd( hnd );
            MEMUnlockHnd( obj );
            throw;
        }

        MEMUnlockHnd( hnd );
        MEMUnlockHnd( obj );
    }
    else
        hnd = NULL;
    raise_if( !hnd, scERRmem );

    memRecordTrackInfo( hnd, filename, line );
    return hnd;
}

/* ======================================================================== */

#endif /* SCDEBUG */

/* ======================================================================== */

scAutoUnlock::scAutoUnlock( scMemHandle hnd )
    : fHandle(hnd)
{
    MEMLockHnd( fHandle );
}

scAutoUnlock::~scAutoUnlock()
{
    MEMUnlockHnd( fHandle );
}

/* ====================================================================== */

#ifndef SCmemset        // we are in a 16 bit world

void scFar*  scFar scCDecl SCmemset( void scFar*    ptr,
                                     int            val,
                                     long           len )
{
    return _fmemset( ptr, val, (size_t)len );
}

/* ====================================================================== */

void scFar*  scFar scCDecl SCmemmove( void scFar*       dst,
                                      const void scFar* src,
                                      long              len )
{
    return _fmemmove( dst, src, (size_t)len );
}

/* ====================================================================== */
```

```
    memRecordTrackInfo( hnd, filename, line);

    dbgTrackAmount( sz + sizeof(MacHandle) );

    return hnd;
}

/* ========================================================================= */

void* MEMResizePtrDebug( void**         obj,
                         ulong          reqSize,
                         const char*    file,
                         int            line )
{
    void        *ptr;

    dbgTrackAmount( reqSize - (int)_msize( *obj ) );

    ptr = realloc( *obj, reqSize );
    raise_if( !ptr, scERRmem );

    return *obj = ptr;
}

/* ========================================================================= */

scMemHandle MEMResizeHndDebug( scMemHandle  obj,
                               ulong        reqSize,
                               const char*  file,
                               int          line )
{
    int size1 = 0;

    if ( obj )
        size1 = _msize( obj );

    if ( !obj )
        obj = MEMAllocHndDebug( reqSize, file, line );
    else {
        scAssert( ((MacHandle*)obj)->Count() == 0 ); // don't resize a locked handle
        obj = (scMemHandle)realloc( obj, reqSize + sizeof( MacHandle ) );
    }

    MacHandle macHandle( obj );

    *(MacHandle*)obj = macHandle;

    int size2 = _msize( obj ) - sizeof( MacHandle );

    dbgTrackAmount( reqSize - size1 );
    return obj;
}

/* ========================================================================= */

void *MEMDupPtrDebug( void *obj, const char *filename, int line )
{
    void        *ptr;

    if ( !RandomFailure() ) {
        ulong       sz = _msize( obj );

        ptr = MEMAllocPtrDebug( sz, filename, line );
        raise_if( !ptr, scERRmem );

        SCmemcpy( ptr, obj, sz );
    }
    else
        ptr = NULL;
    raise_if( !ptr, scERRmem );
    memRecordTrackInfo(ptr, filename, line);
    return ptr;
```

```
void    MEMValidate( void *ptr )
{
}

/* ======================================================================== */

void memDumpMetrics()
{
}

/* ======================================================================== */

inline void memRecordTrackInfo( void *ptr, const char *filename, int line )
{
}

/* ======================================================================== */

inline void memRecordTrackInfo( scMemHandle ptr, const char *filename, int line )
{
}

/* ======================================================================== */

int gRandomFailure;                  // randomly fail memory allocations

static Boolean RandomFailure()
{
    if ( !gRandomFailure )
        return false;

    if ( ( rand() % gRandomFailure ) )
        return false;
    else {
        SCDebugTrace( 0, scString( "RANDOM FAILURE %d\n" ), gRandomFailure );
        return true;
    }
}

/* ======================================================================== */

void* MEMAllocPtrDebug( ulong sz, const char *filename, int line )
{
    void*   ptr;

    raise_if( RandomFailure(), scERRmem );

    scAssert( sz > 0 );
    ptr = malloc( sz );
    raise_if( !ptr, scERRmem );

    memRecordTrackInfo(ptr, filename, line);

    dbgTrackAmount( sz );

    return ptr;
}

/* ======================================================================== */

scMemHandle MEMAllocHndDebug( ulong sz, const char *filename, int line )
{
    scMemHandle hnd;

    raise_if( RandomFailure(), scERRmem );

    hnd = (scMemHandle)malloc( sizeof(MacHandle) + sz );
    raise_if( !hnd, scERRmem );

    MacHandle macHandle( hnd );

    *(MacHandle*)hnd = macHandle;
    ((MacHandle*)hnd)->Validate();
```

```
#endif /* !SCDEBUG */

/* ====================================================================== */

void MEMFreePtr( void* obj )
{

    if ( obj == 0 )
        return;

    dbgTrackAmount( -(int)_msize( obj ) );

    free( obj );
}

/* ====================================================================== */

void MEMFreeHnd( scMemHandle hnd )
{

    if ( hnd == 0 )
        return;

    MacHandle* mh = (MacHandle*)hnd;
    scAssert( !mh->Count() );

    dbgTrackAmount( -(int)_msize( hnd ) );

    free( hnd );
}

/* ====================================================================== */

ulong MEMGetSizePtr( const void *obj )
{
    if ( obj == 0 )
        return 0;

    return _msize( (void*)obj );
}

/* ====================================================================== */

ulong MEMGetSizeHnd( scMemHandle obj )
{
    if ( obj == 0 )
        return 0;

    return _msize( (void*)obj ) - sizeof( MacHandle );
}

/* ====================================================================== */

void *MEMLockHnd( scMemHandle hnd, int counted )
{
    MacHandle* mh = (MacHandle*)hnd;
    return mh->Lock();
}

/* ====================================================================== */

void MEMUnlockHnd( scMemHandle hnd, int counted )
{
    MacHandle* mh = (MacHandle*)hnd;
    mh->Unlock();
}

/* ====================================================================== */

#ifdef MEM_DEBUG

/* ====================================================================== */
```

```
    try {
        void*   srcP = MEMLockHnd( obj );
        void*   dstP = MEMLockHnd( hnd );
        SCmemcpy( dstP, srcP, sz );
    }
    catch( status e ) {
        MEMUnlockHnd( hnd );
        MEMUnlockHnd( obj );
        throw( e );
    }
    catch ( ... ) {
        MEMUnlockHnd( hnd );
        MEMUnlockHnd( obj );
        throw;
    }

    MEMUnlockHnd( hnd );
    MEMUnlockHnd( obj );

    return hnd;
}

/* ========================================================================= */

void *MEMDupObj( void *obj )
{
    void        *ptr;
    ulong       sz = MEMGetSizePtr( obj );

    ptr = MEMAllocPtr( sz );
    raise_if( !ptr, scERRmem );

    SCmemcpy( ptr, obj, sz );
    return ptr;
}

/* ========================================================================= */

void* MEMResizePtr( void**  obj, ulong reqSize )
{
    void        *ptr;

    if ( !*obj )
        ptr = malloc( reqSize );
    else
        ptr = realloc( *obj, reqSize );

    raise_if( !ptr, scERRmem );

    return *obj = ptr;
}

/* ========================================================================= */

scMemHandle MEMResizeHnd( scMemHandle obj, ulong reqSize )
{
    if ( !obj )
        obj = MEMAllocHnd( reqSize );
    else {
        scAssert( ((MacHandle*)obj)->Count() == 0 ); // don't resize a locked handle
        obj = (scMemHandle)realloc( obj, reqSize + sizeof( MacHandle ) );
    }

    MacHandle macHandle( obj );

    *(MacHandle*)obj = macHandle;

    return obj;
}

/* ========================================================================= */
```

```
    #define dbgTrackAmount( n )
#endif

/* ================================================================== */
/* ================================================================== */
/* ================================================================== */

void MEMInit( scPoolInfo [] )
{
}

void MEMFini()
{
}

/* ================================================================== */
/* ================================================================== */
/* ================================================================== */

#ifndef MEM_DEBUG

/* ================================================================== */

void *MEMAllocPtr( ulong sz )
{
    void        *ptr;

    ptr = malloc( sz );
    raise_if( !ptr, scERRmem );

    return ptr;
}

/* ================================================================== */

scMemHandle MEMAllocHnd( ulong sz )
{
    scMemHandle hnd = 0;

    hnd = (scMemHandle)malloc( sizeof( MacHandle) + sz );
    raise_if( !hnd, scERRmem );

    MacHandle macHandle( hnd );

    *(MacHandle*)hnd = macHandle;

    return hnd;
}

/* ================================================================== */

void *MEMDupPtr( void *obj )
{
    void        *ptr;
    ulong       sz = MEMGetSizePtr( obj );

    ptr = MEMAllocPtr( sz );
    raise_if( !ptr, scERRmem );

    SCmemcpy( ptr, obj, sz );
    return ptr;
}

/* ================================================================== */

scMemHandle MEMDupHnd( scMemHandle obj )
{
    scMemHandle hnd;


    ulong   sz = MEMGetSizePtr( obj );

    hnd = MEMAllocHnd( sz );
```

```
/***************************************************************************

    File:      MEM.C

    $Header: /Projects/Toolbox/ct/SCNSHMEM.CPP 2     5/30/97 8:45a Wmanis $

    Contains:  Memory management routines based on our own heap managers

    Written by: Sealy

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

***************************************************************************/

#include "scmem.h"

#include <string.h>
#include <malloc.h>

class  MacHandle {
public:
            MacHandle( scMemHandle  ptr ) :
                    block_( (char*)ptr + sizeof( MacHandle ) ),
                    magic_( 0xfafafafa ),
                    count_( 0 ),
                    size_( _msize( ptr ) - sizeof( MacHandle ) )
                    {
                    }
    void*   Lock( void )
                {
                    Validate();
                    scAssert( count_ >= 0 );
                    count_++;
                    return (void*)block_;
                }
    void    Unlock( void )
                {
                    Validate();
                    scAssert( count_ > 0 );
                    --count_;
                }
    void    Validate()
                {
                    unsigned int size = _msize( this );
                    scAssert( size_ + sizeof( MacHandle ) == size );
                    scAssert( block_ == (char*)this + sizeof( MacHandle ) );
                }
    inline int Count() const
                {
                    return count_;
                }
private:
    const void* block_;
    ulong       magic_;
    int         count_;
    int         size_;
};


#ifdef MEM_DEBUG
    #include <stdlib.h>  // for rand

    long gMemUsage;
    #define dbgTrackAmount( n )     gMemUsage += (n)
#else
```

```
#if defined( MEM_DEBUG )
    return MEMAllocPtrDebug( size, __FILE__, __LINE__ );
#else
    return MEMAllocPtr( size );
#endif
}

/*======================================================================*/

#if defined( MEM_DEBUG )
void *scObject::operator new( size_t        objSize,
                              const char*   file,
                              int           line )
{
    return MEMAllocPtrDebug( objSize, file, line );
}
#endif

/* ====================================================================== */

void scObject::operator delete( void* objStorage )
{
    MEMFreePtr( objStorage );
}

/* ====================================================================== */

int scObject::IsEqual( const scObject& ) const
{
        // if i am all the way down here what can i check, classnames seems
        // a bit late for that
    return true;
}

/* ================================================================== */

int scObject::operator==( const scObject& obj ) const
{
    return IsEqual( obj );
}

/* ================================================================== */

int scObject::operator!=( const scObject& obj ) const
{
    return !IsEqual( obj );
}

/* ================================================================== */
```

```
/*==================================================================*/

scClassInit::scClassInit ( scClass * c )
{
    c->fNext    = c->sClasses;
    c->sClasses = c;
}

/*==================================================================*/

scClass scSimpleObject::sClass =
{
    "scSimpleObject",
    sizeof (scSimpleObject),
    scEmptyClassInitFunc,
    NULL,
    1,
    NULL
};

static scClassInit scSimpleObjectInitClass ( & scSimpleObject::sClass );

/*==================================================================*/

const scClass & scSimpleObject::GetClass ( void ) const
{
    return sClass;
}

/*==================================================================*/

Bool scSimpleObject::IsClass ( const scClass & c ) const
{
    return &c == &sClass;
}

/*==================================================================*/

scClass scObject::sClass =
{
    "scObject",
    sizeof (scObject),
    scEmptyClassInitFunc,
    NULL,
    0,
    NULL
};

static scClassInit scObjectInitClass( & scObject::sClass );

/*==================================================================*/

const scClass & scObject::GetClass( void ) const
{
    return sClass;
}

/*==================================================================*/

Bool scObject::IsClass( const scClass& c ) const
{
    if ( c.IsSimple() )
        return 0;
    for ( const scClass * pc = & GetClass(); pc; pc = pc->GetBase() )
        if ( & c == pc )
            return 1;
    return 0;
}

/*==================================================================*/

void* scObject::operator new( size_t size )
{
```

```
/***************************************************************************

     Stonehand Base Object System Implementation

     $Header: /Projects/Toolbox/ct/SCOBJECT.CPP 2     5/30/97 8:45a Wmanis $

     Contains:

     Written by: Adams

     Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
     All rights reserved.

     This notice is intended as a precaution against inadvertent publication
     and does not constitute an admission or acknowledgment that publication
     has occurred or constitute a waiver of confidentiality.

     Composition Toolbox software is the proprietary
     and confidential property of Stonehand Inc.

***************************************************************************/


#include "scmem.h"
#include "scobject.h"
#include "scexcept.h"

/*=======================================================================*/

const scClass * scClass::sClasses    = NULL;

/*=======================================================================*/

void * scClass::MakeInstance ( void ) const
{
    void *volatile  p = NULL;

    try {
#if defined( MEM_DEBUG )
        p = MEMAllocPtrDebug( fSize, __FILE__, __LINE__ );
#else
        p = MEMAllocPtr( fSize );
#endif
        InitInstance ( (void *) p );
    }
    catch( ... ) {
        MEMFreePtr( p ), p = NULL;
    }
    return p;
}

/*=======================================================================*/

void scClass::InitInstance ( void * p ) const
{
    if ( fInitializer && ( fInitializer != scEmptyClassInitFunc ) )
        ( * fInitializer ) ( p );
    else
        raise( scERRexception );
}

/*=======================================================================*/

const scClass * scClass::FindClass ( const char * className )
{
    if ( ! className )
        return NULL;
    for ( const scClass * pc = sClasses; pc; pc = pc->fNext )
        if ( strcmp ( className, pc->fName ) == 0 )
            return pc;
    return NULL;
}
```

```
#endif /* _H_SBBASE_ */
```

```
    const char*             GetClassname ( void ) const         { return GetClass().GetName(); }

                                // rtti support
    virtual const scClass&  GetClass ( void ) const;
    Bool                    IsClass ( const scClass & c ) const;
    Bool                    IsClass ( const char* name ) const  { return IsClass( *scClass::FindClas
s( name ) ); }

    virtual int             IsEqual( const scObject& ) const;
    int                     operator==( const scObject& ) const;
    int                     operator!=( const scObject& ) const;

 // MEMBERS
public:
    static scClass          sClass;

#if SCDEBUG > 1
    virtual void        DebugPrint( const char * ) const { };
#endif
};


///////////////////////////////////////////////////////////////////////////////
//                             Runtime Type Macros                           //
///////////////////////////////////////////////////////////////////////////////

#define scRTTI(className)   (className::sClass)

#define scDECLARE_RTTI                                          \
        public:                                                 \
           virtual const scClass & GetClass ( void ) const;     \
        public:                                                 \
           static scClass      sClass;                          \
        private:                                                \
           static void         InitInstance ( void * )

#define _scRTTI(className,baseClassName,initFunc,simple) \
        scClass className::sClass = \
            {\
                #className, sizeof(className), initFunc, &scRTTI(baseClassName), simple, 0 }; \
                static scClassInit className##InitClass ( &scRTTI(className) ); \
                const scClass & className::GetClass ( void ) const { return sClass;\
            }

#define scDEFINE_ABSTRACT_RTTI(className,baseClassName) \
        _scRTTI(className,baseClassName,NULL,false)

#define scDEFINE_RTTI(className,baseClassName) \
        void className::InitInstance ( void * p ) { (void) new ( p ) className; } \
        _scRTTI(className,baseClassName,&className::InitInstance,false)

#define scDECLARE_SIMPLE_RTTI \
        public: \
            const scClass &     GetClass ( void ) const; \
            Bool                IsClass ( const scClass & ) const; \
        public: \
            static scClass      sClass; \
        private: \
            static void         InitInstance ( void * );

#define scDEFINE_SIMPLE_RTTI(className,baseClassName) \
        void className::InitInstance ( void * p ) { (void) new ( p ) className; } \
        Bool className::IsClass ( const scClass & c ) const { return &c == &sClass; } \
        _scRTTI(className,baseClassName,&className::InitInstance,true)



#if defined( MEM_DEBUG )
    #define SCNEW   new( __FILE__, __LINE__ )
#else
    #define SCNEW   new
#endif
```

```
    size_t          GetSize ( void ) const      { return fSize; }
    const scClass*  GetBase ( void ) const      { return fBaseClass; }

    Bool            IsAbstract ( void ) const   { return fInitializer == NULL; }
    Bool            IsSimple ( void ) const     { return fSimple; }

    const scClass*  GetNext ( void ) const      { return fNext; }

    static const scClass*   GetClasses ( void ) { return sClasses; }
    static const scClass*   FindClass ( const char * );

// private:
    void            InitInstance ( void * ) const;

 // MEMBERS
    const char*     fName;          // class name
    size_t          fSize;          // instance size
    scClassInitFunc fInitializer;   // instance initializer
    const scClass*  fBaseClass;     // base class
    Bool            fSimple;        // true => non-virtual
    const scClass*  fNext;          // link in class list

    static const scClass*   sClasses;   // class list

};

#define scEmptyClassInitFunc    ( (scClassInitFunc) -1 )

////////////////////////////////////////////////////////////////////////////////
//                              sbSimpleObject                               //
////////////////////////////////////////////////////////////////////////////////

class scSimpleObject {
    // METHODS
public:
                // allocator support
    void*           operator new( size_t s )          { return ::operator new ( s ); }
    void*           operator new ( size_t, void* p )  { return p; }
    void            operator delete ( void* p )       { ::operator delete( p ); }

                // rtti support
  const scClass&    GetClass ( void ) const;
  Bool              IsClass ( const scClass & c ) const;

  // MEMBERS
public:
    static scClass  sClass;
};

////////////////////////////////////////////////////////////////////////////////
//                                  scObject                                 //
////////////////////////////////////////////////////////////////////////////////

class scObject {
 // METHODS
protected:
                        scObject(){}
private:
                            scObject( const scObject & );   // no def
    const scObject&         operator=( const scObject & );  // no def

public:
    virtual             ~scObject(){}



                                // allocator support
    void*                   operator new ( size_t size );
#if defined( MEM_DEBUG )
    void*                   operator new ( size_t size, const char*, int );
#endif
    void*                   operator new ( size_t, void* p )    { return p; }
    void                    operator delete ( void *p );
```

```
/****************************************************************************

    Stonehand Base Object Classes

    $Header: /Projects/Toolbox/ct/SCOBJECT.H 2     5/30/97 8:45a Wmanis $

    Contains:

    Written by: Adams

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

****************************************************************************/


#ifndef _H_SCOBJECT
#define _H_SCOBJECT


#include "sctypes.h"

///////////////////////////////////////////////////////////////////////////////
//                              scClassInit                                 //
///////////////////////////////////////////////////////////////////////////////

class scClass;

class scClassInit {
public:
            scClassInit ( scClass * );
};

///////////////////////////////////////////////////////////////////////////////
//                               scClass                                    //
///////////////////////////////////////////////////////////////////////////////

class scObject;
class scSimpleObject;


typedef void ( * scClassInitFunc ) ( void * );

class scClass {

 // FRIENDS
    friend          scClassInit;

 // METHODS
public:
    static scSimpleObject*  MakeSimpleInstance( const char* name )
                        {
                            const scClass* cl = scClass::FindClass( name );
                            return cl ? (scSimpleObject*)cl->MakeInstance() : 0;
                        }

    static scObject*        MakeInstance( const char* name )
                        {
                            const scClass* cl = scClass::FindClass( name );
                            return cl ? (scObject*)cl->MakeInstance() : 0;
                        }


    void*               MakeInstance ( void ) const;

    const char*     GetName ( void ) const      { return fName; }
    Mark( scREBREAK );
    ForceRepaint( start, end );
    fCharArray.Transform( start, end, chTranType, end - start );
}

#endif

/* ==================================================================== */

int scContUnit::operator==( const scContUnit& p2 ) const
{
    if ( GetContentSize() != p2.GetContentSize() )
        return 0;

    if ( fSpecRun != p2.fSpecRun )
        return 0;

    return fCharArray == p2.fCharArray;
}

/* ==================================================================== */
/* ================================================================== */

Bool scContUnit::FindString( const UCS2*    findString,
                             const SearchState& flags,
                             long&          startOffset,
                             long&          endOffset )
{
    stUnivString ustr;
    ustr.ptr = findString;
    ustr.len = CharacterBufLen( findString );

    if ( fCharArray.FindString( ustr, flags, startOffset, endOffset, startOffset ) ) {
        endOffset = startOffset + ustr.len;
        return true;
    }
    return false;
}

/* ================================================================== */

#ifdef _RUBI_SUPPORT

Bool scContUnit::GetAnnotation( int             nth,
                                long            start,
                                long            end,
                                scAnnotation&   annotation )
{
    if ( fRubiArray ) {
        scRubiData  rd;
        int         index;

        if ( fRubiArray->GetNthRubi( index, rd, nth, start, end ) ) {
            annotation.Set( rd.fCh, GetCount(), rd.fStartOffset, rd.fEndOffset );
            return true;
        }
    }
    return false;
}

/* ================================================================== */

void scContUnit::ApplyAnnotation( long          start,
                                  long          end,
                                  const scAnnotation&   annot )
{
    eChTranType chTranType  = eNormalTran;

    if ( !fRubiArray ) {
        AllocRubiArray();
    }
    else if ( fRubiArray->IsRubiData( start, end ) ) {
        int         nth;
        int         index;
        scRubiData  rd;

        fCharArray.Transform( start, end, eRemoveJapTran, end - start );
        for ( nth = 1; fRubiArray->GetNthRubi( index, rd, nth, start, end );  )
            fRubiArray->RemoveDataAt( index );
    }

    if ( annot.fAnnotate ) {
        scRubiData rd( annot.fCharStr, start, end, SpecAtOffset( start + 1 ) );

        fRubiArray->AddRubiData( rd );

        chTranType = eRubiTran;
    }
    else {
            // i should have already removed any annotations
        if ( fRubiArray->GetNumItems() == 0 )
            DeleteRubiArray();
        chTranType = eRemoveJapTran;
    }
    fSpecRun.DebugRun( "ReplaceToken" );
#endif
    return 1;
}

/* ==================================================================== */

int scContUnit::GetToken( stUnivString& ustr,
                          int32         start,
                          int32         end ) const
{
    return fCharArray.GetToken( ustr, start, end );
}

/* ==================================================================== */
// memory comes in here locked

Bool scContUnit::ReplaceWord( CharRecordP&      startChRec,
                              scSpecRecord*&    specRec,
                              long              startOffset,
                              long&             endOffset,
                              long&             limitOffset,
                              UCS2*             chBuf,
                              UCS2*             replaceBuf )
{
    long    deltaLen;

    deltaLen = CharacterBufLen( replaceBuf ) - CharacterBufLen( chBuf );

    if ( !deltaLen ) {
        LoadNewWord( startChRec + startOffset,
                     replaceBuf,
                     endOffset - startOffset );
        return false;
    }

        // need to grow or shrink memory
    if ( deltaLen > 0 ) // grow the memory - first resize and then move
        PARASetChSize( p, startChRec, deltaLen );


    memmove( *startChRec + *endOffset + deltaLen,
             *startChRec + *endOffset,
             (size_t)( p->fChSize - *endOffset + 1 ) * sizeof( CharRecord ) );
    p->fChSize      += deltaLen;
    *limitOffset    += deltaLen;
    *endOffset      += deltaLen;

    scAssert( (*startChRec + p->fChSize)->character == 0 );

//  if ( deltaLen < 0 ) // shrink the memory - first move and then resize
//      PARASetChSize(  p, startChRec, deltaLen );


    LoadNewWord( startChRec + startOffset, replaceBuf, endOffset - startOffset );

    fSpecRun.BumpOffset( endOffset - deltaLen, deltaLen );
    specRec = fSpecRun.ptr( );

    return true;
}
/* ==================================================================== */

int scContUnit::FindString( const stUnivString& ustr,
                            const SearchState&  flags,
                            int32               start,
                            int32               end,
                            int32&              offset )
{
    return fCharArray.FindString( ustr, flags, start, end, offset );
}
```

```
void scContUnit::Iter( SubstituteFunc   func,
                       long             startLocation,
                       long&            limitOffset )
{
    UCS2            chBuf[64];
    UCS2*           chP;
    CharRecordP     startChRec;
    scSpecRecord*   specRec;
    long            startOffset,
                    endOffset,
                    wordLen;

    LockMem( startChRec, specRec );

    startOffset = startLocation;
    endOffset   = startOffset;

    for ( ; endOffset < limitOffset; ) {
        startOffset = TXTStartSelectableWord( startChRec, endOffset );
        endOffset   = TXTEndSelectableWord( startChRec, endOffset );
        wordLen = endOffset - startOffset;

        if ( wordLen > 1 ) {
            BuildTestWord( chBuf, startChRec + startOffset, wordLen );
            status stat = (*func)( &chP, chBuf, NULL );

            if ( stat == scSuccess || stat == scUserAbort ) {
                if ( !ReplaceWord( startChRec, specRec, startOffset, endOffset,
                                   limitOffset, chBuf, chP ) ) {
                    UnlockMem( );
                    return;
                }

                fCharArray.RepairText( fSpecRun, startOffset, endOffset );
                Mark( scREBREAK );

                if ( stat == scUserAbort )
                    goto exit;
            }
        }
        endOffset = FindNextSpellingWord( startChRec, endOffset, limitOffset );
    }

exit:
    UnlockMem( );
}

/*********************************************************************/

int scContUnit::ReplaceToken( const stUnivString& ustr,
                              int32               start,
                              int32&              end )
{
    ForceRepaint( start, end );
    Mark( scREBREAK );

    if ( ustr.len == (ulong)(end - start) && fCharArray.ReplaceToken( ustr, start, end ) ) {
        fCharArray.RepairText( fSpecRun, start, end );
    }
    else if ( fCharArray.Insert( ustr, start, end ) ) {
        int32   diff = ( ustr.len - ( end - start ) );
        fSpecRun.BumpOffset( start, diff );

        end += diff;
        TypeSpec ts;
        fCharArray.Retabulate( fSpecRun, start, end, ts,
                               fCharArray.GetContentSize() );
    }
    else {
        scAssert( 0 );
    }
#if SCDEBUG > 0
    fCharArray.Validate();
```

```
/*=====================================================================

    File:       scparag2.c

    $Header: /Projects/Toolbox/ct/Scparag2.cpp 4     5/30/97 8:45a Wmanis $

    Contains:   content unit implementations

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

=====================================================================*/


#include "scparagr.h"
#include "scctype.h"
#include "scspcrec.h"
#include "scmem.h"
#include "scannota.h"

#ifdef _RUBI_SUPPORT
#include "scrubi.h"
#endif

#define INLINE static

/*************************************************************************/

INLINE void LoadNewWord( CharRecordP    ch,
                         const UCS2*    replaceCh,
                         long           size )
{
    for ( ; size--; )
        (ch++)->character = *replaceCh++;
}

/*************************************************************************/

INLINE void BuildTestWord( UCS2*        ch,
                           CharRecordP  charRec,
                           long         size )
{

    for ( ; size--; )
        *ch++ = (charRec++)->character;
    *ch = 0;
}

/*************************************************************************/

INLINE long FindNextSpellingWord( CharRecordP   startChRec,
                                  long          endOffset,
                                  long          limitOffset )
{
    UCS2 ch = startChRec[endOffset].character;

    while ( !CTIsAlpha( ch ) && endOffset < limitOffset )
        ch = startChRec[endOffset++].character;
    return endOffset;
}

/*************************************************************************/
```

```
                    testGetStrip = false;

                if ( !breakControl && cData.ResetOrphan( testGetStrip ) ) {
                    breakControl++;
                    return eRebreak;
                }

                cData.fCol->DeleteExcessLines( this, cData.fPData.fPrevline, testGetStrip, cData );
                testGetStrip = true;

                    // find the next column
                if ( cData.FindNextCol( dcState ) ) {
                        // the finding of the next column may reset the para spec
                    scCachedStyle::SetParaStyle( this, defspec_ );
                    continue;
                }
                else {
                    overFlow = true;
                    break;
                }
            }


        }

        tryAgain = ResetWidow( cData, testGetStrip );

    }

    if ( GetFirstline() )
        cData.PARADeleteExcessLines( );

#if SCDEBUG > 1
    SetReformatEvent( reformatEvent++ );
#endif

    Unmark( scREBREAK );

    if ( !overFlow )
        scAssert( GetLastline() != 0 );

    cData.fPData.PARAFini( );

    prevParaData.lastLineH  = cData.fPData.fTextline;
    prevParaData.lastSpec   = cData.fPData.fCurSpecRec->spec();

    SCDebugTrace( 1, scString( "scContUnit::Reformat OUT 0x%08x %d\n" ), this, GetCount() );

    return !overFlow ? eNormalReformat : eOverflowGeometry;
}

/* ==================================================================== */
```

```
                                )
{

    DCState dcState;
    Bool    tryAgain;
    Bool    leadRetry       = false;    // a retry based upon a leading increase on the line
    Bool    overFlow        = false;
    Bool    testGetStrip    = true;


    SCDebugTrace( 1, scString( "scContUnit::Reformat IN 0x%08x %d\n" ), this, GetCount() );

    cData.PARAInit( this, breakControl, keepWNextControl, prevParaData );

    dcState.SetColumn( cData.GetActive()  );
    scFlowDir fd( cData.fCol->GetFlowdir() );

    if ( cData.fPData.fPrevline ) {
        if ( fd.IsHorizontal() )
            cData.fSavedPrevEnd.x = LONG_MAX;
        else {
            cData.fSavedPrevEnd.y = LONG_MAX;
        }
    }

    for ( tryAgain = true; tryAgain; ) {
        for ( overFlow = false; cData.fPData.fBreakType != eEndStreamBreak || overFlow; ) {

            if ( !overFlow )
                cData.fPData.SetLineData( leadRetry );      // set up initial line conditions

            if ( !overFlow && cData.AllocGeometry() ) {     // allocatate the geometry from
                                                            // the parent column, if we overflow
                                                            // the column, we go to the next column

                try {
                    cData.AllocLine( leadRetry );           // allocate the memory, reuse scheme is
used
                }
                catch( ... ) {
                    cData.fPData.PARAFini( );
                    throw;
                }

                cData.fPData.SetColumn( cData.fCol );
                leadRetry = cData.fPData.ComposeLine( dcState );  // compose the line

                if ( !leadRetry ) {
                    // we are accepting the line as is
                    MicroPoint x, y;
                    if ( fd.IsHorizontal() ) {
                        x = cData.fPData.fComposedLine.fOrg.x + cData.fPData.fComposedLine.fMeasure;
                        y = cData.fPData.fComposedLine.fOrg.y;
                    }
                    else {
                        x = cData.fPData.fComposedLine.fOrg.x;
                        y = cData.fPData.fComposedLine.fOrg.y + cData.fPData.fComposedLine.fMeasure;
                    }
                    cData.fSavedPrevEnd.Set( x, y );
                }
                else
                    ; // we are going to reposition the line and rebreak it
            }
            else {
                    // No more room in column, let's try the next column
                overFlow    = false;

                    // if we try and relead at the bottom of a container and we
                    // overflow we need to set "leadRetry" to false since at this
                    // point it is not a retry any more but a whole new container
                leadRetry   = false;

                if ( !cData.fPData.fPrevline )
r() ) {
            /* each time we try to remove just one line */
            /* back up to previous column */
        cData.COLFini( false );
        cData.fCol  = cData.fCol->GetPrev();
        cData.COLInit( cData.fCol, this );

            /* remove one excess line */
        cData.fPData.fPrevline = cData.fCol->GetLastline();
        if ( cData.fPData.fPrevline ) {
            cData.fPData.fPrevline = LNPrev( cData.fPData.fPrevline );
            cData.fCol->DeleteExcessLines( this, cData.fPData.fPrevline, testGetStrip, cData );

                /* reset state */
            if ( cData.fPData.fPrevline ) {
                cData.fPData.fLinesBefore                   = cData.fPData.fPrevline->GetLinecount()
;
                cData.fPData.fLineNumber                    = (short)(cData.fPData.fLinesBefore + 1)
;
                cData.fPData.fInitialLine.fStartCharOffset  = cData.fPData.fPrevline->GetEndOffset()
;

                cData.fPData.fCurSpecRec                  = cData.fPData.GetSpecRecord( cData.fPData.f
Prevline->GetEndOffset() );
                TypeSpec ts = cData.fPData.fCurSpecRec->spec();
                scCachedStyle::GetCachedStyle( ts );
                cData.fPData.fBreakType                 = eColumnBreak;

                return true;
            }
        }
    }
    return false;
}

/* ================================================================== */
/* perform the line breaking on the paragraph

eRefEvent scContUnit::Reformat( scCOLRefData&   cData,
                                PrevParaData&   prevParaData,
                                int             keepWNextControl
#if SCDEBUG > 1
                                , int&          reformatEvent
#endif
                                )

    int         breakControl = 0;
    eRefEvent   refEvent;
    do {

        refEvent = Reformat2( cData,
                              prevParaData,
                              keepWNextControl,
                              breakControl
#if SCDEBUG > 1
                              , reformatEvent
#endif
                               );

    } while ( refEvent == eRebreak );
    return refEvent;
}

/* ================================================================== */

eRefEvent scContUnit::Reformat2( scCOLRefData&  cData,
                                 PrevParaData&  prevParaData,
                                 int            keepWNextControl,
                                 int&           breakControl
#if SCDEBUG > 1
                                , int&          reformatEvent
#endif
```

```
    if ( keepWNextControl ) {
        fPData.fPrevline            = NULL;
        fPData.fLinesBefore         = 0;
        fPData.fLinesAfter          = 0;
        fPData.fColumnCount         = 0;
    }
    else if ( breakControl ) {
        fPData.fPrevline                = NULL;
    }
    else {
        fPData.fPrevline = fPData.fPara->LocateFirstLine( *this,
                                                          fPData.fCurSpecRec->spec(),
                                                          fCol,
                                                          fPData.fComposedLine.fBaseline,
                                                          fPData.fComposedLine.fEndLead,
                                                          prevParaData );

        fPData.fLinesBefore     = 0;
        fPData.fLinesAfter      = 0;
        fPData.fColumnCount     = 0;
    }

        // this is where in a layout world I would make the distinction between
        // a logical unit and a paragraph
    fPData.fComposedLine.fLastLineLen   = LONG_MIN;
}

/* ================================================================== */
/* Check for orphan or no break condition violation. If found, delete   */
/* excess lines of last column and return true so para reformat can     */
/* try again. If pData.lastTxlH is NULL, there are no lines in the      */
/* column, and we did not fail due to an orphan condition -- then,      */
/* delete all lines in column.                                          */

Bool scCOLRefData::ResetOrphan( Bool testGetStrip )
{
    scContUnit*p = fPData.GetPara();
    scAssert( p != 0 );

    if ( fPData.fBreakParams.NoBreak() || fPData.fLinesBefore < fPData.fBreakParams.LinesBefore() )


        if ( p->GetPrev() && fCol->GetNext() ) {

            fPData.fPrevline    = p->GetPrev()->GetLastline();
            fCol->DeleteExcessLines( p, fPData.fPrevline, testGetStrip, *this );
            fPData.PARAFini( );
            COLFini( true );

            fCol = fCol->GetNext();
            return COLInit( fCol, p );
        }
    }

    return false;
}

/* ================================================================== */
/* Check here for widows. If the columns had the same measure we could  */
/* just grab sufficient lines, but since columns may be of different    */
/* measure, we will just grab one line at a time and check the fit.     */
/* Each time we iterate, we will grab another line. This is slower but  */
/* more acurate. An optimization may be to check the measures and, if   */
/* they are close, grab more than one line on an iteration.             */

/* If a widow is found, reset parameters and return true so             */
/* para reformat can try again.                                         */

Bool scContUnit::ResetWidow( scCOLRefData&  cData,
                             Bool           testGetStrip )
{
    if ( cData.fPData.fColumnCount && cData.fPData.fLinesAfter < cData.fPData.fBreakParams.LinesAfte
    fLinesBefore++;
    if ( fColumnCount )
        fLinesAfter++;
    fPrevline = fTextline;

    return false;
}

/* ====================================================================== */
/* This column is full. If there is a next one, get it ready for          */
/* reformatting and return true; else, return false;                      */

Bool scCOLRefData::FindNextCol( DCState& dcState )
{
    Bool colRefStat = false;

    if ( fCol->GetNext() ) {
        COLFini( true );
        fCol         = fCol->GetNext();
        colRefStat   = COLInit( fCol, fPData.fPara );

            // if the character count is zero we are transitioning
            // columns before we set any text so reset the dcState column
        if ( fPData.fComposedLine.fCharCount == 0 )
            dcState.SetColumn( fCol  );
    }

    if ( colRefStat == true ) {
        fPData.fPrevline                    = NULL;
        fPData.fTextline                    = NULL;
        fPData.fInitialLine.fLastLineLen    = LONG_MIN;
        fPData.fBreakType                   = eCharBreak;
        fPData.fColumnCount++;
        fPData.fLinesAfter                  = 0;
        return true;
    }

    fPData.fTextline = fPData.fPrevline;
    return false;
}

/* ====================================================================== */
/* Set up initial paragraph reformat data.                                */

void scCOLRefData::PARAInit( scContUnit*    p,
                             int            breakControl,
                             int            keepWNextControl,
                             PrevParaData&  prevParaData )
{
    fPData.PARAInit( p, fCol->GetFlowdir() );

    fPData.fPara->SetFirstline( 0 );

    fPData.fCurSpecRec          = fPData.fStartSpecRec;

    fPData.fTextline            = NULL;
    fPData.fBreakType           = eParaBreak;
    fPData.fLineNumber          = 0;
    fPData.fLinesHyphed         = 0;


    fPData.fInitialLine.Init( fCol->GetFlowdir() );
    fPData.fInitialLine.SetCharacters( fPData.fCharRecs );

    fPData.fComposedLine.Init( fCol->GetFlowdir() );
    fPData.fComposedLine.SetCharacters( fPData.fCharRecs );

    scCachedStyle::GetParaStyle().GetParaBreak( fPData.fBreakParams );

    if ( fPData.fBreakParams.KeepWithNext() )
         fPData.fPara->Mark( scKEEPNEXT );
    else
         fPData.fPara->Unmark( scKEEPNEXT );
    }
    else if ( fBreakLang == eCompJapanese ) {
        fBreakType = BRKJapanLineBreak( fComposedLine.fCharRecs,
                                        fComposedLine.fStartCharOffset,
                                        fComposedLine.fCharCount,
                                        fComposedLine,
#ifdef scUseRubi
                                        fAnnotations,
#endif
                                        fLineNumber,
                                        fLinesHyphed,
                                        &fCurSpecRec,
                                        fComposedLine.fInkExtents,
                                        fComposedLine.fLetterSpace,
                                        dcState );
        scAssert( fComposedLine.fInkExtents.Valid() );
    }
#endif

    if ( AdjustLead() ) {
        fBreakType = eUndefinedBreak;
        fCurSpecRec = curSpecRec;
        if ( fInitialLine.fBaseline == FIRST_LINE_POSITION ) {
            TypeSpec ts = fInitialLine.GetInitialSpec();
            MicroPoint firstline = CSfirstLinePosition( GetColumn()->GetAPPName(), ts );
            fInitialLine.fBaseline = fComposedLine.fBaseline - firstline;
        }
        return true;
    }

    fTextline->Set( fLineNumber, fBreakType, fComposedLine );

    if ( !fPara->GetFirstline() ) {
        scAssert( fTextline->GetEndOffset() <= fPara->GetContentSize() );
        scAssert( fTextline->GetStartOffset() == 0 );
        fPara->SetFirstline( fTextline );
    }

    scLEADRefData   aboveLeadData( fComposedLine.fInitialLead.GetFlow() );
    scLEADRefData   belowLeadData( fComposedLine.fInitialLead.GetFlow() );

    if ( fComposedLine.GetMaxLeadSpec() != fCurSpecRec->spec() ) {
        TypeSpec ts = fComposedLine.GetMaxLeadSpec();
        scCachedStyle::GetCachedStyle( ts );
        belowLeadData.ComputeAboveBelow( scCachedStyle::GetCurrentCache().GetComputedLead(), scCache
dStyle::GetCurrentCache().GetFlowdir() );
        TypeSpec ts1 = fCurSpecRec->spec();
        scCachedStyle::GetCachedStyle( ts1 );
        aboveLeadData.ComputeAboveBelow( scCachedStyle::GetCurrentCache().GetComputedLead(), scCache
dStyle::GetCurrentCache().GetFlowdir() );
    }
    else {
        TypeSpec ts = fCurSpecRec->spec();
        scCachedStyle::GetCachedStyle( ts );
        belowLeadData.ComputeAboveBelow( scCachedStyle::GetCurrentCache().GetComputedLead(), scCache
dStyle::GetCurrentCache().GetFlowdir() );

        int endoffset = fTextline->GetEndOffset() + 1;
        if ( endoffset >= (fCurSpecRec+1)->offset() ) {
            fCurSpecRec++;
            ts = fCurSpecRec->spec();
            scCachedStyle::GetCachedStyle( ts );
        }
        aboveLeadData.ComputeAboveBelow( scCachedStyle::GetCurrentCache().GetComputedLead(), scCache
dStyle::GetCurrentCache().GetFlowdir() );
    }

    fComposedLine.fEndLead.SetBelowLead( belowLeadData.GetBelowLead() );
    fComposedLine.fEndLead.SetAboveLead( aboveLeadData.GetAboveLead() );
    fComposedLine.fEndLead.SetExternalSpace( 0 );


    fLineNumber++;
```

```
        // reset things
    fComposedLine                       =  fInitialLine;
}

/* ==================================================================== */

void scPARARefData::PARAInit( scContUnit*p,
                              const scFlowDir& fd )
{
    scAssert( fPara == 0 );

    fPara           = p;
    fCharRecs       = (CharRecordP)fPara->GetCharArray().Lock( );

    fStartSpecRec   = fPara->GetSpecRun().ptr( );

    TypeSpec ts = fStartSpecRec->spec();
    fPara->InitParaSpec( ts );

    fComposedLine.Init( fd );

        // !!!!!NOTE: this should only be set here  or in the line breaker
    fComposedLine.SetMaxLeadSpec( fStartSpecRec->spec() );


    scCachedStyle::GetCachedStyle( ts );

    fBreakLang      = scCachedStyle::GetCurrentCache().GetBreakLang();
}

/* ==================================================================== */

void scPARARefData::PARAFini(  )
{
    fPara->GetCharArray().Unlock( );

    fPara           = 0;
    fCharRecs       = 0;
    fStartSpecRec   = 0;

}

/* ==================================================================== */
// Break and set the current line of the paragraph,
//          RETURNS     false if no need to retry the linebreaker
//                      true if we need to retry due to a leading change
//

Bool scPARARefData::ComposeLine( DCState& dcState )
{
    scSpecRecord*   curSpecRec = fCurSpecRec;

        // set primary lead - for ???
    fComposedLine.fInitialLead.SetAboveLead( 0 );
    fComposedLine.fInitialLead.SetBelowLead( 0 );

    scAssert( fComposedLine.fCharRecs != 0 );
    scAssert( fComposedLine.GetMaxLeadSpec() != 0 );

#ifdef scJIS4051
    if ( fBreakLang == eCompRoman ) {
#endif
        fBreakType = BRKRomanLineBreak( fComposedLine.fCharRecs,
                                        fComposedLine.fStartCharOffset,
                                        fComposedLine.fCharCount,
                                        fComposedLine,
                                        fLineNumber,
                                        fLinesHyphed,
                                        &fCurSpecRec,
                                        fComposedLine.fInkExtents,
                                        fComposedLine.fLetterSpace );
        scAssert( fComposedLine.fInkExtents.Valid() );
#ifdef scJIS4051
```

```
        fPData.fTextline->InitForReuse( fPData.fPara );

    return fPData.fTextline != NULL;
}

/* ==================================================================== */
// Set up line data according to current spec

void scPARARefData::SetLineData( Bool leadRetry )
{
    if ( leadRetry ) {
            // we want to reset the initial leading and the max lead spec
            // but everything else should be the same
            // because we are going to retry setting the line
        fInitialLine.fEndLead.SetBelowLead( fInitialLine.fInitialLead.GetBelowLead() );

        MicroPoint maxAboveLead = MAX( fInitialLine.fEndLead.GetAboveLead(),
                                       fComposedLine.fEndLead.GetAboveLead() );
        fInitialLine.fEndLead.SetAboveLead( maxAboveLead );
        fInitialLine.SetMaxLeadSpec( fComposedLine.GetMaxLeadSpec() );
        fComposedLine           = fInitialLine;
    }

    scAssert( fInitialLine.fCharRecs != 0 );

        // we have accepted the composed line at this point
        // so we can copy it over
    fInitialLine                        = fComposedLine;

    fInitialLine.fStartCharOffset       = fComposedLine.GetEndCharOffset();
    fInitialLine.fCharCount             = 0;
    fCharCount

        // get to the spec at the beginning of the line
    fInitialLine.fSpecRec = GetSpecRecord( fInitialLine.fStartCharOffset );
    TypeSpec ts = fInitialLine.fSpecRec->spec();
    fInitialLine.SetInitialSpec( ts );
    scCachedStyle::GetCachedStyle( ts );

    if ( !leadRetry )
        fInitialLine.SetMaxLeadSpec( ts );

    fInitialLine.fStartSpecRunOffset    = 0;
    fInitialLine.fSpecRunCount          = 0;

#ifdef scUseRubi
    fInitialLine.fAnnotations           = 0;
#endif

    fInitialLine.fInkExtents            =  scCachedStyle::GetCurrentCache().GetInkExtents();
    fInitialLine.fInkExtents.Translate( 0, scCachedStyle::GetCurrentCache().GetBaseline() );
    fInitialLine.fLogicalExtents        =  scCachedStyle::GetCurrentCache().GetLogicalExtents();

// fLastLineLen
// fMeasure
// fComputedLen
// fRagSetting


        // set primary lead - we are setting here for COLGetStrip
    fInitialLine.fInitialLead           = fComposedLine.fEndLead;
// fEndLead
// fLetterSpace

// fColShapeType

    fInitialLine.fBaselineJump          =  scCachedStyle::GetCurrentCache().GetBaseline();

// fFlowDir

        // now the the initial values are set copy them over to the line
        // that we will compose, if the composition fails for some
        // reason or other we will have our initial values and can
    if ( GetPrev() ) {    /* there are prior paragraphs in the stream */
        lastTxl = GetPrev()->GetLastline();

        if ( lastTxl ) {
            lastTxl->ParaLead( lead, col->GetFlowdir() );

            if ( lastTxl->GetColumn() != cData.GetActive() ) {
                cData.COLFini( false );
                col              = lastTxl->GetColumn();
                cData.COLInit( col, this );
            }
            baseline        = lastTxl->GetBaseline();

            if ( col->GetFlowdir().IsHorizontal() )
                cData.fPrevEnd.Set( col->Width(), lastTxl->GetOrigin().y );
            else
                cData.fPrevEnd.Set( lastTxl->GetOrigin().x, col->Depth() );
            cData.fSavedPrevEnd = cData.fPrevEnd;

                // we have to fool the baseline into thinking that it is
                // in a vertically oriented column
            if ( col->GetFlowdir().IsVertical() )
                baseline = col->Width() - baseline;
        }
        else {
                // overflow stuff, in COLGetStrip
            TypeSpec ts = SpecAtStart( );
            lead.Set( scCachedStyle::GetCachedStyle( ts ).GetComputedLead(), col->GetFlowdir() );
            baseline    = LONG_MIN;
        }
    }
    else {
            // this is the first paragraph in the stream
        TypeSpec ts = SpecAtStart(  );
        lead.Set( scCachedStyle::GetCachedStyle( ts ).GetComputedLead(), col->GetFlowdir() );
        lastTxl     = NULL;
        baseline    = FIRST_LINE_POSITION;
    }
    return lastTxl;
}

/* ===================================================================== */

Bool scCOLRefData::AllocLine( Bool leadRetry )
{
    if ( leadRetry ) {
        scAssert( fPData.fTextline != 0 );
        return true;
    }

    if ( !fPData.fPrevline ) {
        fPData.fTextline = fCol->GetFirstline();
        if ( fPData.fTextline && fPData.fTextline->GetPara()->GetCount() < fPData.fPara->GetCount()
) {
            fCol->FreeLines( true, fLineDamage );
            fPData.fTextline = NULL;
        }
    }
    else
        fPData.fTextline = fPData.fPrevline->GetNext();

    if ( !fPData.fTextline ) {
        fPData.fTextline = scTextline::Allocate( fPData.fPara,
                                                 fCol,
                                                 fPData.fPrevline );
    }
    else if ( fPData.fTextline->GetPara() != fPData.fPara ) {
        fPData.fPrevline = fPData.fTextline;
        fPData.fTextline = LNInsertNew( fPData.fPara, fCol, fPData.fTextline );
    }
    else
```

```
{
//  TypeSpec maxLeadSpec    = fLineData.fMaxLeadSpec;

//  fBreakType              = fSavedBreakType;
//  fCurSpec                = fSavedCurSpec;
//  fLineData               = fSavedLineData;
//  cData                   = fSavedscCOLRefData;

//  fLineData.fMaxLeadSpec  = maxLeadSpec;

//  FMSetMetrics( fCurSpec );
}

/* ==================================================================== */

Bool scPARARefData::AdjustLead( void ) const
{
    return fInitialLine.fInitialLead.GetAboveLead() < fComposedLine.fEndLead.GetAboveLead();
}

/* ==================================================================== */
// delete excess lines in the paragraph we are currently formatting

void scCOLRefData::PARADeleteExcessLines( void )
{
    scContUnit* para    = fPData.fPara;
    scTextline* txl     = fPData.fTextline;
    scTextline* nextTxl;
    scColumn*   nextCol;

    if ( txl )
        txl = txl->GetNext();
    else
        txl = para->GetFirstline();

        // delete excess lines in paragraph
    for ( ; txl; txl = nextTxl ) {

        nextTxl = txl->GetNext();

        if ( nextTxl == NULL ) {
                // check next column
            if ( ( nextCol = txl->GetColumn()->GetNext() ) != NULL )
                nextTxl = para->NextColumn( nextCol );
        }

        if ( txl->GetPara() == para )
            txl->Delete( fLineDamage );
        else
            break;
    }

        // delete any excess lines in column if no further paragraphs
    if ( para && !para->GetNext() && txl ) {
        for ( ; txl; txl = nextTxl ) {
            nextTxl = LNNext( txl );
            txl->Delete( fLineDamage );
        }
    }

}

/* ==================================================================== */
/* figure out where to put the first line of a paragraph */

scTextline* scContUnit::LocateFirstLine( scCOLRefData&  cData,
                                          TypeSpec       curSpec,
                                          scColumn*&     col,
                                          MicroPoint&    baseline,
                                          scLEADRefData& lead,
                                          PrevParaData&  prevParaData )
{
    scTextline* lastTxl;
```

```
        case kRomanBaseline:
            fd.SetFlow( eRomanFlow );
            break;

            // vertical flow
        case kLeftBaseline:
        case kCenterBaseline:
        case kRightBaseline:
            fd.SetFlow( eVertJapanFlow );
            break;
    }
    ComputeAboveBelow( lead, fd );
    return GetLead();
}

/* ==================================================================== */

void scLEADRefData::Set( MicroPoint aboveLead, MicroPoint belowLead, const scFlowDir& fd )
{
    fAboveLead  = aboveLead;
    fBelowLead  = belowLead;
    fFlow       = fd;
}

/* ==================================================================== */

void scLEADRefData::ComputeAboveBelow( MicroPoint lead, const scFlowDir& fd )
{
    if ( fd.IsHorizontal() ) {
        static REAL realAbove = (REAL)RLU_BASEfmTop / scBaseRLUsystem;
        static REAL realBelow = (REAL)RLU_BASEfmBottom / scBaseRLUsystem;
        static MicroPoint lastlead;
        static MicroPoint abovelead;
        static MicroPoint belowlead;

        if ( lastlead != lead ) {
            abovelead   = scRoundMP( realAbove * lead );
            belowlead   = scRoundMP( realBelow * lead );
            lastlead    = lead;
        }
        fAboveLead = abovelead;
        fBelowLead = belowlead;
    }
    else {
        fAboveLead  = lead / 2;
        fBelowLead  = lead / 2;
    }
}

/* ==================================================================== */

scSpecRecord* scPARARefData::GetSpecRecord( long offset )
{
    scSpecRecord* specrec;

    for ( specrec = fStartSpecRec; offset > (specrec+1)->offset(); specrec++ )
        ;
    return specrec;
}

/* ==================================================================== */

void scPARARefData::SaveData( /* const scCOLRefData& cData */ )
{
//  fSavedBreakType     = fBreakType;
//  fSavedLineData      = fLineData;
//  fSavedscCOLRefData  = cData;
//  fSavedCurSpec       = fCurSpec;
}

/* ==================================================================== */

void scPARARefData::RestoreData( /* scCOLRefData& cData */ )
```

```
    fOrg.Set( 0, 0 );

    fCharRecs       = 0;
    fStartCharOffset= 0;
    fCharCount      = 0;

    fSpecRec        = 0;
    fStartSpecRunOffset = 0;
    fSpecRunCount   = 0;

    // fMaxLeadSpec; DO NOT SET THIS IN HERE

#ifdef _RUBI_SUPPORT
    fAnnotations    = 0;
#endif

    fInkExtents.Invalidate();
    fLogicalExtents.Invalidate();

    fLastLineLen    = 0;
    fMeasure        = 0;
    fComputedLen    = 0;
    fRagSetting     = eRagCentered;
    fLetterSpace    = 0;
    fStartAngle     = 0;
    fEndAngle       = 0;
    fColShapeType   = eNoShape;
    fBaselineJump   = 0;
}

/* ================================================================== */

void scLINERefData::Init( const scFlowDir& fd )
{
    xxInit();
    fInitialLead.Init( fd );
    fEndLead.Init( fd );
    fFlowDir        = fd;
}

/* ================================================================== */

scLEADRefData::scLEADRefData( const scLEADRefData& ld )
{
    fFlow       = ld.fFlow;
    fAboveLead  = ld.fAboveLead;
    fBelowLead  = ld.fBelowLead;
}

/* ================================================================== */

void scLEADRefData::Set( MicroPoint lead )
{
    ComputeAboveBelow( lead, fFlow );
}

/* ================================================================== */

void scLEADRefData::Set( MicroPoint lead, const scFlowDir& fd )
{
    ComputeAboveBelow( lead, fd );
}

/* ================================================================== */

MicroPoint scLEADRefData::Compute( MicroPoint ptsize, MicroPoint lead, eFntBaseline baseline )
{
    scFlowDir fd;
    switch ( baseline ) {
            // horizontal flow
        case kTopBaseline:
        case kMiddleBaseline:
        case kBottomBaseline:
```

```
/*================================================================

    File:       scparag3.c

    $Header: /Projects/Toolbox/ct/SCPARAG3.CPP 3     5/30/97 8:45a Wmanis $

    Contains:   reformatting code for content units.

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.


================================================================*/


#include "scparagr.h"
#include "sccolumn.h"
#include "scbreak.h"
#include "scstcach.h"
#include "scglobda.h"
#include "scctype.h"
#include "scmem.h"
#include "scspcrec.h"
#include "scstream.h"
#include "sctextli.h"
#include "screfdat.h"
#include "scparagr.h"
#include "sccolumn.h"
#include "sccallbk.h"

#ifdef scUseRubi
#include "scrubi.h"
#endif

/* ================================================================== */

scPARARefData::scPARARefData() :
    fPrevPara( 0 ),
    fPrevSpec( 0 ),
    fOrigin( 0, 0 ),
    fPrevline( 0 ),
    fTextline( 0 ),
    fPara( 0 ),
    fBreakType( eUndefinedBreak ),
    fColumnCount( 0 ),
    fLinesBefore( 0 ),
    fLinesAfter( 0 ),
    fLineNumber( 0 ),
    fLinesHyphed( 0 ),
    fCharRecs( 0 ),
    fStartSpecRec( 0 ),
    fCurSpecRec( 0 ),
    fSpecCount( 0 ),
#ifdef scUseRubi
    fAnnotations( 0 ),
#endif
    column_( 0 )
{
}

/* ================================================================== */

void scLINERefData::xxInit( )
{
 */

scContUnit* scContUnit::Earlier( const scContUnit* p2 ) const
{
    const scContUnit* prevPara = this;

    for ( ; prevPara != NULL; prevPara = prevPara->GetPrev( ) )
        if ( prevPara == p2 )
            return (scContUnit*)p2;

    return (scContUnit*)this;
}

/* ==================================================================== */

scContUnit* scContUnit::Allocate( TypeSpec&     spec,
                                  scContUnit*   cu,
                                  long          ct )
{
    return SCNEW scContUnit( spec, cu, ct );
}

/* ==================================================================== */
```

09653370.072400

```
    int i = fSpecRun.indexAtOffset( startOffset );
    do {
        TypeSpec ts = fSpecRun[i].spec();
        tsList.Insert( ts );
    } while ( fSpecRun[++i].offset() < endOffset );
}

/* ====================================================================== */
/* insert specs in this the paragraph */

void scContUnit::GetTSList( scTypeSpecList& tsList )
{
    OffsetGetTSList( LONG_MIN, LONG_MAX, tsList );
}

/* ====================================================================== */

void scContUnit::OffsetGetCharSpecList( long            startOffset,
                                        long            endOffset,
                                        scSpecLocList&  csList )
{
    if ( endOffset == startOffset )
        return;

    if ( endOffset == LONG_MAX )
        endOffset = GetContentSize();
    if ( startOffset == LONG_MIN )
        startOffset = 0;

    int i = fSpecRun.indexAtOffset( startOffset );
    scSpecLocation chsploc( GetCount(), startOffset );
    chsploc.spec() = fSpecRun[i].spec();
    csList.Append( chsploc );

    while ( fSpecRun[++i].offset() < endOffset ) {
        scSpecLocation chsploc( GetCount(), fSpecRun[i].offset() );
        chsploc.spec() = fSpecRun[i].spec();
        csList.Append( chsploc );
    }

    csList.TermParagraph( GetCount(), endOffset );
}

/* ====================================================================== */

void scContUnit::SelectWord( long   offset,
                             long&  startWord,
                             long&  endWord )
{
    fCharArray.SelectWord( offset, startWord, endWord );
}

/* ====================================================================== */

void scContUnit::Deformat( )
{
    scTextline* txl;
    scTextline* next;

    for ( txl = GetFirstline(); txl && txl->GetPara() == this; txl = next ) {
        next = txl->GetNextLogical();
        if ( txl && txl->GetColumn() )
            txl->GetColumn()->Mark( scINVALID );
        txl->MarkForDeletion( );
    }
    SetFirstline( 0 );
    Mark( scREBREAK );
}

/* ====================================================================== */
/* return the earlier of these two paragraphs in a stream
 * NULL would indicate that they are not in the same stream
```

```
}

/* ================================================================== */
// unlink a paragraph from a stream

void scContUnit::Unlink( )
{
    scContUnit* lastPara;
    scContUnit* nextPara;

        // mark all the lines of the paragraph as being invalid
    Deformat( );

    lastPara = GetPrev();
    nextPara = GetNext();

    if ( lastPara )
        lastPara->SetNext( nextPara );
    else {
            // this is the first paragraph in the stream and we must let
            // the columns know that the head of the stream has changed
            //

        GetStream()->ResetStream( (scStream*)nextPara );

    }
    if ( nextPara )
        nextPara->SetPrev( lastPara );

    SetPrev( NULL );
    SetNext( NULL );
}

/* ================================================================== */
/* return spec at begginning of paragraph */

TypeSpec scContUnit::SpecAtEnd( )
{
    for ( int i = 0; !fSpecRun[i].isTerminator(); i++ )
        ;
    return fSpecRun[i].spec();
}

/* ================================================================== */
/* return spec at end of paragraph */

TypeSpec scContUnit::SpecAtStart( )
{
    return fSpecRun[0].spec();
}

/* ================================================================== */
/* return spec at offset of paragraph */

TypeSpec scContUnit::SpecAtOffset( long offset )
{
    return fSpecRun.SpecAtOffset( MAX( 0, offset - 1)  );
}

/* ================================================================== */
/* insert specs in this selection of the paragraph */

void scContUnit::OffsetGetTSList( long              startOffset,
                                  long              endOffset,
                                  scTypeSpecList&   tsList )
{
    if ( endOffset == startOffset )
        return;

    if ( endOffset == LONG_MAX )
        endOffset = GetContentSize();
    if ( startOffset == LONG_MIN )
        startOffset = 0;
        if ( txl->GetPara( ) != this )
            break;
        prevTxl = txl;
    }
    return prevTxl;
}

/* ================================================================== */
/* find the last visible paragraph in a stream */

scContUnit* scContUnit::GetLastVisiblePara( ) const
{
    const scContUnit*   lastp;
    const scContUnit*   p = this;

    for ( lastp = p; p; p = p->GetNext() ) {
        if ( !p->GetFirstline() )
            break;
        lastp = p;
    }

    return (scContUnit*)lastp;
}

/* ================================================================== */
/* return the previous visible paragraph */

scContUnit* scContUnit::GetPrevVisiblePara( void ) const
{
    const scContUnit* p = this;

    for ( ; p; p = p->GetPrev() ) {
        if ( p->GetFirstline() )
            break;
    }
    return (scContUnit*)p;
}

/* ================================================================== */
/* renumber the paragraphs of a stream */

void scContUnit::Renumber( )
{
    long        count;
    scContUnit* p = this;

        // back up
    for ( ; p && p->GetPrev(); p = p->GetPrev() )
        ;

        // renumber
    for ( count = 0; p; p = p->GetNext() )
        p->SetCount( count++ );
}

/* ================================================================== */
/* report back size of paragraph on disk */

long scContUnit::ExternalSize( ) const
{
    long exSize;

    exSize = (long)sizeof(scContUnit);

    exSize += fCharArray.ExternalSize();
    exSize += fSpecRun.ExternalSize( );

#ifdef _RUBI_SUPPORT
    if ( GetRubiArray() )
        exSize += GetRubiArray()->ExternalSize();
#endif

    return exSize;
```

```
{
    fCharArray.WriteText( fSpecRun, addDcr, ctxPtr, writeFunc );
}

/* ================================================================== */

void scContUnit::ReadAPPText( stTextImportExport& appText )
{
    fCharArray.ReadAPPText( fSpecRun, appText );
    Mark( scRETABULATE );
}

/* ================================================================== */

void scContUnit::WriteAPPText( stTextImportExport& appText )
{
    appText.StartPara( defspec_ );
    fCharArray.WriteAPPText( fSpecRun, appText );
}

/* ================================================================== */
/* scRETABULATE a paragraph, if ts is NULL whole para will be reformatted
 * otherwise only the ts section will be reformatted
 */

void scContUnit::Retabulate( TypeSpec ts )
{
    if ( GetContentSize() > 0 ) {
        fCharArray.Retabulate( fSpecRun, 0L, GetContentSize(), ts, GetContentSize() );
        Mark( scREBREAK );
    }
    Unmark( scRETABULATE );
}

/* ================================================================== */
/* find the last line of a paragraph */

scTextline* scContUnit::GetLastline() const
{
    scTextline* txl;
    scTextline* nexttxl;

    scAssertValid();
    for ( txl = fFirstline; txl; txl = nexttxl ) {
        if ( txl->IsLastLinePara() )
            break;

        nexttxl = txl->GetNextLogical();

        if ( !nexttxl || nexttxl->GetPara( ) != this )
            break;
    }

    if ( txl )
        scAssert( txl->GetPara( ) == this );
    return txl;
}

/* ================================================================== */
/* find the last visible line of a paragraph */

scTextline* scContUnit::GetLastVisibleLine( ) const
{
    scTextline* prevTxl;
    scTextline* txl;

    scAssertValid();

        // start with first line of paragraph
    prevTxl = fFirstline;

    for ( txl = prevTxl; txl; txl = txl->GetNextLogical() ) {
```

```
{
    scContUnit* p = this;
    scContUnit* nextPara;

    bytesFreed = 0;
    for ( ; p; p = nextPara ) {
        nextPara = p->GetNext( );
        bytesFreed += p->ExternalSize();
        p->Free();
    }
}

/* ================================================================== */
/* this is the simple case of freeing the paragraph,
 * NO disentangling of pointers
 */

void scContUnit::Free( scSelection* select )
{
    if ( select ) {
            // if para is on selection list remove it from selection list
        select->CheckFreePara( this );
    }

#ifdef _RUBI_SUPPORT
        // free rubi if present
    DeleteRubiArray();
#endif

    delete this;
}

/* ================================================================== */
/* duplicate a paragraph, using old paragraph as the model and linking it
/* to 'prevParaH'


scContUnit* scContUnit::Copy( scContUnit* prevPara ) const
{
    TypeSpec nullSpec;

    scContUnit* dstPara = scContUnit::Allocate( nullSpec, prevPara, GetCount() );

    fCharArray.Copy( dstPara->GetCharArray(), 0, fCharArray.GetContentSize() );
    dstPara->CopySpecRun( fSpecRun );
    TypeSpec ts = defspec_;
    dstPara->SetDefaultSpec( ts );

#ifdef _RUBI_SUPPORT
    if ( fRubiArray ) {
        dstPara->AllocRubiArray( *fRubiArray );
    }
#endif

    return dstPara;
}

/* ================================================================== */

long scContUnit::ReadStream( APPCtxPtr  ctxPtr,
                             IOFuncPtr  readFunc )
{
    long    ret = fCharArray.ReadText( fSpecRun, ctxPtr, readFunc );
    Mark( scRETABULATE );

    return ret;
}

/* ================================================================== */

void scContUnit::WriteStream( Bool      addDcr,
                              APPCtxPtr ctxPtr,
                              IOFuncPtr writeFunc )
    }
    else
        fCharArray.CharInfo( fSpecRun, offset, ch, flags, escapement, ts, unitType );

    fCharArray.WordSpaceInfo( offset, wordspace );
}

/* ================================================================== */
/* transform the text style between offset1 and offset2, return the column or
 * previous column containing the effected text
 */

void scContUnit::TextTrans( long        offset1,
                            long        offset2,
                            eChTranType trans,
                            int         numChars )
{
    long    tmp;

    if ( offset1 >= offset2 ) {
        if ( offset1 == offset2 )
            return;
        tmp = offset1;
        offset1 = offset2;
        offset2 = tmp;
    }

    if ( offset1 == LONG_MIN )
        offset1 = 0;
    if ( offset2 == LONG_MAX )
        offset2 = GetContentSize();

    fCharArray.Transform( offset1, offset2, trans, numChars );

    TypeSpec nullSpec;
    fCharArray.Retabulate( fSpecRun, offset1, offset2, nullSpec, GetContentSize() );

    Mark( scREBREAK );
}

/* ================================================================== */

void scContUnit::InitParaSpec( TypeSpec& ts )
{
    if ( !defspec_.ptr() )
        defspec_ = ts;
    scCachedStyle::SetParaStyle( this, defspec_ );
}

/* ================================================================== */

void scContUnit::LockMem( CharRecordP&      chRec,
                          scSpecRecord*&    specRec )
{
    chRec   = (CharRecordP)GetCharArray().Lock( );
    specRec = fSpecRun.ptr();

    TypeSpec ts = specRec->spec();
    InitParaSpec( ts );
    scCachedStyle::GetCachedStyle( ts );
}

/* ================================================================== */

void scContUnit::UnlockMem(  )
{
    GetCharArray().Unlock( );
}

/* ================================================================== */
/* free a scrap handle */

void scContUnit::FreeScrap( long& bytesFreed )
```

```
    long    tmp;
    long    compOffset1,
            compOffset2;

    if ( offset1 >= offset2 ) {
        tmp     = offset1;
        offset1 = offset2;
        offset2 = tmp;
    }

    compOffset1 = MAX( offset1, 0 );
    compOffset2 = MIN( offset2, GetContentSize() );

        // apply the spec only if it is an emopty para
    if ( compOffset1 == 0 && compOffset2 == 0 && GetContentSize() )
        return;

    if ( compOffset1 == 0 || compOffset1 != compOffset2 ) {
        if ( compOffset2 == GetContentSize() )
            fSpecRun.ApplySpec( style, compOffset1, LONG_MAX );
        else
            fSpecRun.ApplySpec( style, compOffset1, compOffset2 );

#ifdef _RUBI_SUPPORT
        if ( fRubiArray )
            fRubiArray->ApplyStyle( compOffset1, compOffset2, style );
#endif

        fSpecRun.SetContentSize( GetContentSize() );
        if ( forceRepaint )
            ForceRepaint( compOffset1, compOffset2 );

        if ( retabulate )
            fCharArray.Retabulate( fSpecRun, compOffset1, compOffset2, style, GetContentSize() );
        else
            Mark( scRETABULATE );

        Mark( scREBREAK );
    }
}

/* ================================================================== */
/* fill in some specific information about a character */

void scContUnit::ChInfo( long           offset,
                         UCS2&          ch,         /* character at offset */
                         ulong&         flags,
                         MicroPoint&    escapement, /* escapement at offset */
                         MicroPoint&    wordspace,  /* ws escapement at offset */
                         TypeSpec&      ts,         /* typespec at offset */
                         eUnitType&     unitType )  /* relative or absolute */
{
    if ( offset == 0 ) {
        ch          = scParaStart;
        flags       = 0;
        escapement  = 0;
        ts          = fSpecRun.SpecAtOffset( offset );
        unitType    = eAbsUnit;
    }
    else if ( offset == GetContentSize() + 1 ) {
        ch          = scParaEnd;
        flags       = 0;
        escapement  = 0;
        ts          = fSpecRun.SpecAtOffset( GetContentSize() );
        unitType    = eAbsUnit;
    }
    else if ( offset > GetContentSize() ) {
        ch          = 0;
        flags       = 0;
        escapement  = 0;
        ts.clear();
        unitType    = eAbsUnit;
        if ( doit )
            fSpecRun.PrintRun( "scContUnit::CharInsert" );
    }
#endif

#ifdef _RUBI_SUPPORT
    if ( fRubiArray ) {
        if ( fRubiArray->IsRubiData( offset + computedOffset ) ) {
            scRubiData rd;
            fRubiArray->GetRubiAt( rd, offset + computedOffset );
            fCharArray.Transform( rd.fStartOffset, rd.fEndOffset, eRemoveJapTran, 0 );

            fRubiArray->DeleteRubiData( offset );
            if ( !fRubiArray->GetNumItems() )
                DeleteRubiArray();
        }
    }
#endif

    if ( computedOffset >= 0 )
        fCharArray.SetNumSlots( fCharArray.GetNumItems() + 1 );

    fCharArray.CharInsert( tmMove,
                           fSpecRun,
#ifdef _RUBI_SUPPORT
                           fRubiArray,
#endif
                           offset,
                           keyRec,
                           textCleared,
                           clearedSpec );

    fSpecRun.SetContentSize( GetContentSize() );

#if SCDEBUG > 1
    {
        static int doit;
        if ( doit )
            fSpecRun.PrintRun( "void scContUnit::CharInsert 2" );
    }
#endif

    scTextline* txl = FindLine( offset );
    if ( txl )
        txl->Mark( scREPAINT ); /* force repaint */

    Mark( scREBREAK );

    rebreak = true;
}

/* ================================================================== */

void scContUnit::SetDefaultSpec( TypeSpec& ts )
{
    if ( ts.ptr() != defspec_.ptr() ) {
        defspec_ = ts;
        Mark( scREBREAK );
    }
    ForceRepaint( 0, LONG_MAX );
}

/* ================================================================== */
/* set the text style between offset1 and offset2, return the column or
 * previous column containing the effected text
 */

void scContUnit::SetStyle( long     offset1,
                           long     offset2,
                           TypeSpec style,
                           Bool     retabulate,
                           Bool     forceRepaint )
{
```

```
                iAmRemoved = true;

                if ( startPara ) {
                    startPara->Mark( scREBREAK );
                    rebreak = true;
                    keyRec.replacedchar() = scParaSplit;
                }
                else
                    keyRec.noop() = true;

                tmMove = 0;

                return startPara;
            }
            else if ( keyRec.keycode() == scForwardDelete && offset == GetContentSize() ) {
                if ( GetNext() == NULL )
                    keyRec.noop() = true;
                else {
                    startPara = GetNext()->Merge( offset );
                    startPara->Mark( scREBREAK );
                    rebreak = true;
                    keyRec.replacedchar() = scParaSplit;
                    keyRec.restoreselect() = true;
                        /* flag to reset cursor behind new ch */
                }

                tmMove = 0;
                break;
            }
            computedOffset = -1;
            /* FALL THROUGH */
        default:
            startPara->CharInsert( computedOffset,
                                   offset,
                                   keyRec,
                                   tmMove,
                                   rebreak,
                                   textCleared,
                                   clearedSpec );
            break;
    }
    return startPara;
}

/* ==================================================================== */

void scContUnit::Insert( const CharRecord&  ch,
                         TypeSpec&          spec,
                         long               offset )
{
    CharRecordP chRec = (CharRecordP)&ch;
    fCharArray.Insert( chRec, offset, 1 );
    fSpecRun.BumpOffset( offset, fCharArray.GetNumItems() );

    if ( spec.ptr() )
        fSpecRun.ApplySpec( spec, offset, offset + 1 );

    Mark( scREBREAK );
}

/* ==================================================================== */

void scContUnit::CharInsert( long        computedOffset,
                             long&       offset,
                             scKeyRecord&   keyRec,
                             long&       tmMove,
                             short&      rebreak,
                             Bool        textCleared,
                             TypeSpec    clearedSpec )
{
#if SCDEBUG > 1
    {
        static int doit;
        case scUpArrow:
            tmMove                  = PREV_LINE;
            keyRec.replacedchar()   = scDownArrow;
            break;
        case scDownArrow:
            tmMove                  = NEXT_LINE;
            keyRec.replacedchar()   = scUpArrow;
            break;
        case scLeftArrow:
            tmMove                  = -1;
            keyRec.replacedchar()   = scRightArrow;
            break;
        case scRightArrow:
            tmMove                  = 1;
            keyRec.replacedchar()   = scLeftArrow;
            break;
    }
}

/* ====================================================================== */

scContUnit* scContUnit::KeySplit( long&         offset,
                                  scKeyRecord&  keyRec,
                                  long&         tmMove,
                                  short&        rebreak )
{
    scContUnit* p = Split( offset );

    offset      = 0;
    rebreak = true;
    if ( keyRec.restoreselect() ) { /* replacing forward deletion */
        tmMove = -1;
        keyRec.replacedchar() = scForwardDelete;
    }
    else {
        tmMove = 0;
        keyRec.replacedchar() = scBackSpace;
    }
    return p;
}

/* ====================================================================== */

scContUnit* scContUnit::KeyInsert( long&            offset,
                                   scKeyRecord&     keyRec,
                                   long&            tmMove,
                                   short&           rebreak,
                                   Bool             textCleared,
                                   TypeSpec         clearedSpec,
                                   Bool&            iAmRemoved )
{
    scContUnit* startPara       = this;
    long        computedOffset  = 0;


        /* insert the character into the text */
    switch ( keyRec.keycode() ) {
        case scUpArrow:
        case scDownArrow:
        case scLeftArrow:
        case scRightArrow:
            ArrowSupport( keyRec, tmMove );
            break;

        case scParaSplit:
            startPara = KeySplit( offset, keyRec, tmMove, rebreak );
            break;

        case scBackSpace:
        case scForwardDelete:

            if ( keyRec.keycode() == scBackSpace && offset == 0 ) {
                startPara = Merge( offset );
            }
            if ( txl->IsLastLinePara( ) )
                break;
        }
    }
}

/* ================================================================== */
/* find the location of 'offset' in this paragraph */

Bool scContUnit::FindLocation( long&        offset,     /* offset location to find */
                               Bool&        endOfLineP,/* true if cursor stays at end of
                                                        * this line instead of moving to
                                                        * next on hyphenated word
                                                        */
                               scTextline*& txl,    /* line that offset is on */
                               MicroPoint&  hLoc,       /* location on line from org*/
                               eContentMovement cursDirect )
{
    scTextline* ntxl;
    Bool        endOfLine   = false;

        // find the first line of the paragraph
    txl = fFirstline;

        // search the lines of the paragraph until we find a line
        // containing the 'offset'
    for ( ; txl; txl = ntxl ) {

        if ( offset >= txl->GetStartOffset ()) {
            if ( offset < txl->GetEndOffset ()) {
                break;                  /* we found it */
            }
                /* Stop here if we are at the end of a hyphenated
                 * line and endOfLineP is true
                 */
            else if ( endOfLineP && offset == txl->GetEndOffset() && txl->IsHyphenated()) {
                endOfLine = true;
                break;                  /* we found it */
            }
        }

        ntxl = txl->GetNextLogical();               /* get next line */
        if ( !ntxl || ntxl->GetPara() != this )
            break;
    }

    endOfLineP  = endOfLine;

    if ( txl ) {
        scMuPoint   charOrg;

            // find location on line
        charOrg = txl->Locate( offset, charOrg, cursDirect );


        if ( txl->GetColumn()->GetFlowdir().IsVertical() )
            hLoc = charOrg.y;
        else
            hLoc = charOrg.x;
        return true;
    }

    hLoc    = LONG_MIN;
    return false;
}

/* ================================================================== */

static void ArrowSupport( scKeyRecord&  keyRec,
                          long&         tmMove )
{
    switch ( keyRec.keycode() ) {
        if ( GetRubiArray() ) {
            GetRubiArray()->DeleteRubiData( offset1, offset2 );
            if ( !GetRubiArray()->GetNumItems() )
                DeleteRubiArray();
        }
#endif
    }

    Mark( scREBREAK );

        /* break link to first line if we remove that text, the refernece
         * to this text will be patched in the reformatting process
         */
    txl             = GetFirstline();

    if ( txl ) {
        scColumn*   col = txl->GetColumn();
        if ( txl && col->GetRecomposition() ) {
            scTextline* nextTxl;

            for ( ; txl && txl->GetPara( ) == this; txl = nextTxl ) {
                nextTxl = txl->GetNextLogical();
                if ( offset2 >= txl->GetEndOffset( ) ) {
                        // the delete takes care of patching the para
                    txl->MarkForDeletion( );
                }
                else {
                    long startOffset = MIN( txl->GetStartOffset( ) - offset2, GetContentSize() );
                    long endOffset   = MIN( txl->GetEndOffset( ) - offset2, GetContentSize() );

                    txl->SetOffsets( startOffset, endOffset );
                }
            }
        }
    }

    return entireParaDeleted;
}

/* ================================================================== */
/* force a repaint of the logical selection - map the logical world
 * into the layout world
 */

void scContUnit::ForceRepaint( long offset1,
                               long offset2 )
{
    scTextline* txl;
    scTextline* ntxl;
    scColumn*   col;

    txl = GetFirstline();

    if ( txl ) {
        col = txl->GetColumn();

            /* search the lines of the paragraph until we find a line
             * containing the 'offset'
             */
        for ( ; txl; txl = ntxl ) {

            if ( offset2 < txl->GetStartOffset( ) )
                break;
            else if ( offset1 > txl->GetEndOffset( ) )
                ;
            else if ( offset1 < txl->GetEndOffset( ) )
                txl->Mark( scREPAINT );

            ntxl = LNNext( txl );
            if ( ntxl == NULL ) {
                    /* hit the bottom of a column, check next column */
                col = col->GetNext();
                ntxl = NextColumn( col );
    ClearText( offset, GetContentSize() );
    if ( offset == 0 )
        SetFirstline( NULL );

    PostInsert( p2 );

    Mark( scREBREAK );
    p2->Mark( scREBREAK );
    offset = 0;
    return p2;
}

/* ================================================================ */
// merge this paraH with the previous, "this" is deleted in
// this method, the new content unit is returned

scContUnit* scContUnit::Merge( long& offset )
{
    scStream*   stream  = (scStream*)FirstInChain();
    scContUnit* prev = GetPrev();

    if ( prev ) {

        scColumn* col = scColumn::FindFlowset( stream );

        offset = prev->GetContentSize();

        Unlink( );
        prev->Renumber();

        long tmp = offset;

        prev->PasteText( this, tmp );

        Free( col ? col->FlowsetGetSelection() : 0 );

        return prev;
    }
    return this;
}

/* ================================================================ */
/* this clears the text from a paragraph and returns true if the entire
 * text of paragraph has been deleted
 */

Bool scContUnit::ClearText( long        offset1,
                            long        offset2 )
{
    scTextline* txl;
    Bool        entireParaDeleted = false;

    offset1 = MAX( MIN( offset1, GetContentSize() ), 0 );
    offset2 = MIN( MAX( offset2, 0 ), GetContentSize() );

    if ( offset1 == 0 && offset2 == GetContentSize() )
        entireParaDeleted = true;

    if ( entireParaDeleted ) {
        GetCharArray().RemoveBetweenOffsets( offset1, offset2 );
        fSpecRun.SetContentSize( 0 );

#ifdef _RUBI_SUPPORT
        DeleteRubiArray();
#endif
    }
    else {
        GetCharArray().RemoveBetweenOffsets( offset1, offset2 );
        fSpecRun.Clear( offset1, offset2 );
        GetCharArray().RepairText( fSpecRun, offset1, offset1 );

#ifdef _RUBI_SUPPORT
```

```
        newPara = srcPara->CopyText( 0L, PARAChSize( (scContUnit*)srcPara ) );
        firstPara->PostInsert( newPara );
        firstPara = newPara;
    }
    tmpOffset = 0;
    finalPara->PasteText( srcPara, tmpOffset );

    offset = srcPara->GetContentSize();
    Renumber();

    return finalPara;
}

/* ================================================================== */
/* paste one para into another para */

void scContUnit::PasteText( const scContUnit*   srcPara,
                            long&               offset )
{
    try {
        if ( offset == 0 ) {
            TypeSpec ts = srcPara->GetDefaultSpec();
            if ( ts.ptr() )
                SetDefaultSpec( ts );
        }

            // paste the specs in
        fSpecRun.InsertRun( offset, srcPara->GetContentSize(), srcPara->GetSpecRun() );

            // paste the text in
        fCharArray.Paste( ((scContUnit*)srcPara)->GetCharArray(), offset );
        Mark( scREBREAK );

#ifdef _RUBI_SUPPORT
            // paste the rubis in
        if ( fRubiArray || srcPara->GetRubiArray() ) {

            if ( fRubiArray && !srcPara->GetRubiArray() )
                fRubiArray->BumpRubiData( offset, srcPara->GetContentSize() );
            else if ( !fRubiArray && srcPara->GetRubiArray() ) {
                AllocRubiArray( *srcPara->GetRubiArray() );
                fRubiArray->BumpRubiData( 0, offset );
            }
            else
                fRubiArray->Paste( *srcPara->GetRubiArray(), offset, srcPara->GetContentSize() );
        }
#endif

        scTextline* txl = FindLine( offset );
        if ( txl )
            txl->Mark( scREPAINT ); /* force repaint */

        long startOffset = offset;
        offset += srcPara->GetContentSize();

        fSpecRun.SetContentSize( GetContentSize() );

        fCharArray.RepairText( fSpecRun, startOffset, offset );
    }
    catch( ... ) {
        SCDebugBreak(); // remove stuff from the paragraph
        throw;
    }
}

/* ================================================================== */
/* split the paragraph into to two paragraphs at the split */

scContUnit* scContUnit::Split( long& offset )
{
    scContUnit* p2;

    p2 = CopyText( offset, GetContentSize() );
```

```
        if ( txl ) {
            if ( txl->GetPara( ) != this )
                txl = NULL;
            break;
        }
    }
    return txl;
}

/* ====================================================================== */

scTextline* scContUnit::FindLine( long offset ) const
{
    scTextline* txl  = fFirstline;

    for ( ; txl && txl->GetPara() == this; txl = txl->GetNextLogical() ) {
        if ( txl->OffsetOnLine( offset ) )
            return txl;
    }
    return NULL;
}

/* ====================================================================== */
/* append para2H to the stream containing para1H */

void scContUnit::Append( scContUnit* p2 )
{
    scContUnit* p1 = (scContUnit*)LastInChain();
    p1->SetNext( p2 );
    p2->SetPrev( p1 );
    Renumber();
}

/* ====================================================================== */
/* insert para2 into the stream following para1 */

void scContUnit::PostInsert( scContUnit *p2 )
{
    scContUnit *nextP;

    if ( p2 ) {
        nextP = GetNext( );
        SetNext( p2 );
        p2->SetPrev( this );

        scContUnit* last = (scContUnit*)p2->LastInChain();

        last->SetNext( nextP );
        if ( nextP )
            nextP->SetPrev( last );
        Renumber( );
    }
}

/* ====================================================================== */

scContUnit* scContUnit::PasteParas( const scContUnit*   srcPara,
                                    long&               offset )
{
    scContUnit* finalPara;
    scContUnit* newPara;
    scContUnit* firstPara = this;
    long        tmpOffset;

        // split the para
    finalPara = Split( offset );

    tmpOffset = GetContentSize();
    PasteText( srcPara, tmpOffset );

    srcPara = srcPara->GetNext( );

    for ( ; srcPara->GetNext( ); srcPara = srcPara->GetNext( ) ) {
```

```
/* ================================================================== */

void scContUnit::DebugParaSpecs()
{
    SCDebugTrace( 0, scString( "\npara spec - 0x%08x %d\n" ), defspec_, GetContentSize() );
    fSpecRun.PrintRun( "para specs" );
}

#endif

/* ================================================================== */

scContUnit::scContUnit() :
    fFirstline( 0 ),
    fParaCount( 0 )
{
}

/* ================================================================== */

scContUnit::scContUnit( TypeSpec&   spec,
                        scContUnit* prevPara,
                        long        count ) :
                                fFirstline( 0 ),
                                fParaCount( 0 ),
                                fSpecRun( spec ),
                                defspec_( spec )
#ifdef _RUBI_SUPPORT
                                , fRubiArray( 0 )
#endif
{
#if SCDEBUG > 1
    fReformatEvent = 0;
#endif

    try {
        SetCount( count );

        Mark( scRETABULATE );

        if ( prevPara )
            prevPara->SetNext( this );
        SetPrev( prevPara );
    }
    catch( ... ) {
        delete this;
        throw;
    }
}

/* ================================================================== */

scContUnit::~scContUnit()
{
//  SCDebugTrace( 0, scString( "scContUnit::~scContUnit: 0x%08x\n" ), this );
}


/* ================================================================== */
/* find the next line in a following column containing
 * lines of this paragraph
 */

scTextline* scContUnit::NextColumn( scColumn*& col )
{
    scTextline* txl = NULL;

        // check next column
    for ( ; col; col = col->GetNext() ) {
        if ( col )
            txl = col->GetFirstline( );
    if ( offset2 < offset1 ) {
        long tmp;
        tmp     = offset1;
        offset1 = offset2;
        offset2 = tmp;
    }

    offset2 = MIN( offset2, GetContentSize() );
    offset1 = MAX( 0, offset1 );

    long maxSpecRecs    = fSpecRun.NumItems() + dstPara->fSpecRun.NumItems();
    dstPara->fSpecRun.SetNumSlots( maxSpecRecs );

    long maxChars       = fCharArray.GetNumItems() + dstPara->GetCharArray().GetNumItems();
    dstPara->GetCharArray().SetNumSlots( maxChars );

        // copy specrun
    fSpecRun.Copy( dstPara->fSpecRun, offset1, offset2 );

        // copy the text
    fCharArray.Copy( dstPara->GetCharArray(), offset1, offset2 );

#ifdef _RUBI_SUPPORT
        // copy the rubi annotations
    if ( fRubiArray ) {
        dstPara->AllocRubiArray( *GetRubiArray() );

        dstPara->GetRubiArray()->DeleteRubiData( offset2, GetContentSize() );


        dstPara->GetRubiArray()->DeleteRubiData( 0, offset1 );

        if ( !dstPara->GetRubiArray()->GetNumItems() )
            dstPara->DeleteRubiArray();
    }
#endif

    dstPara->GetCharArray().RepairText( dstPara->fSpecRun, 0L, dstPara->GetContentSize() );

//  dstPara->fSpecRun.SetContentSize( dstPara->GetContentSize() );

    return dstPara;
}

/* ==================================================================== */

#if SCDEBUG > 1
void scContUnit::DbgPrintInfo( int debugLevel ) const
{
    SCDebugTrace( debugLevel, scString( "\nSCPARAGRAPH  - reformat event %d\n" ), fReformatEvent );
    SCDebugTrace( debugLevel, scString( "firstline 0x%08x count %d\n" ), fFirstline, fParaCount );
    SCDebugTrace( debugLevel, scString( "retabulate %u rebreak %u\n" ), fLogBits.fRetabulate, fLogBı
ts.fRebreak );
    SCDebugTrace( debugLevel, scString( "reposition %u logactive %u\n" ), fLogBits.fReposition, fLog
Bits.fLogActive );
    SCDebugTrace( debugLevel, scString( "keepnext %u\n" ), fLogBits.fKeepNext );
}

/* ==================================================================== */

void scContUnit::scAssertValid( Bool recurse ) const
{
    scTBObj::scAssertValid();

    scAssert( !Marked( scRETABULATE ) );
    scAssert( !Marked( scREBREAK ) );

    if ( fFirstline ) {
        fFirstline->scAssertValid( false );
        scAssert( fFirstline->GetStartOffset() == 0 );
        scAssert( fFirstline->GetEndOffset() <= fCharArray.GetContentSize() );
    }
}
```

```
/* ================================================================== */

Bool scContUnit::IsRubiPresent( size_t start, size_t end )
{
    if ( !fRubiArray )
        return false;

    return fRubiArray->IsRubiData( start, end );
}

/* ================================================================== */

#endif

scColumn* scContUnit::GetFirstCol( void ) const
{
    return fFirstline ? fFirstline->GetColumn() : 0;
}

/* ================================================================== */
// mark the paragraph

void scContUnit::Mark( const scLogBits& bits )
{
    scTBObj::Mark( bits );

    SCDebugTrace( 1, scString( "scContUnit::Mark 0x%08x %d\n" ), this, GetCount() );

    if ( bits.fRetabulate || bits.fRebreak || bits.fReposition ) {
        if ( fFirstline ) {
            scColumn* col = fFirstline->GetColumn();
            col->Mark( scINVALID );
        }
    }
}

/* ================================================================== */
// do this paragraph and this column intersect

Bool scContUnit::ColSect( const scColumn* col ) const
{
    const scTextline *l;

        // if we use this maybe this method should be moved over to column
    for ( l = col->GetFirstline(); l; l = l->GetNext() ) {
        if ( l->GetPara() == this )
            return true;
    }
    return false;
}

/* ================================================================== */
// does this para contain this spec

Bool scContUnit::ContainTS( TypeSpec ts )
{
    if ( ts == 0 )      // 0 being the short cut for all specs
        return true;
    else if ( ts == GetDefaultSpec() )
        return true;

    return fSpecRun.Includes( ts );
}

/* ================================================================== */

scContUnit* scContUnit::CopyText( long offset1, long offset2 ) const
{
    TypeSpec defspec = defspec_;

    scContUnit* dstPara = scContUnit::Allocate( defspec );
#endif

    Mark( scRETABULATE );
    Mark( scREBREAK );
}

/* ==================================================================== */

void scContUnit::Write( APPCtxPtr   ctxPtr,
                        IOFuncPtr   writeFunc )
{
    scTBObj::Write( ctxPtr, writeFunc );

    WriteLong( fParaCount, ctxPtr, writeFunc, kIntelOrder );

        // the characters
    fCharArray.Write( ctxPtr, writeFunc );

        // the spec runs
    fSpecRun.Write( ctxPtr, writeFunc );

    WriteLong( APPPointerToDiskID( ctxPtr, defspec_.ptr(), diskidTypespec ),
               ctxPtr,
               writeFunc,
               kIntelOrder );

#ifdef _RUBI_SUPPORT
    WriteLong( (ulong)fRubiArray ? 1 : 0, ctxPtr, writeFunc, kIntelOrder );
    if ( fRubiArray )
        fRubiArray->Write( ctxPtr, writeFunc );
#endif
}

/* ==================================================================== */

void scContUnit::RestorePointers( scSet* enumTable )
{
    if ( !Marked( scPTRRESTORED ) ) {
        scTBObj::RestorePointers( enumTable );
        fSpecRun.RestorePointers( );
    }
}

/* ==================================================================== */

void scContUnit::CopySpecRun( const scSpecRun& spr )
{
    fSpecRun = spr;
}

/* ==================================================================== */

#ifdef _RUBI_SUPPORT

void scContUnit::AllocRubiArray( void )
{
    fRubiArray = SCNEW scRubiArray;
}

/* ==================================================================== */

void scContUnit::AllocRubiArray( const scRubiArray& ra )
{
    fRubiArray = SCNEW scRubiArray;
    *fRubiArray = ra;
}

/* ==================================================================== */

void scContUnit::DeleteRubiArray( void )
{
    delete fRubiArray, fRubiArray = 0;
}
```

```
/**************************************************************************

    File:      SCPARAGR.C

    $Header: /Projects/Toolbox/ct/SCPARAGR.CPP 2     5/30/97 8:45a Wmanis $

    Contains:   content unit code.

    Written by: Manis

    Copyright (c) 1989-94 Stonehand Inc., of Cambridge, MA.
    All rights reserved.

    This notice is intended as a precaution against inadvertent publication
    and does not constitute an admission or acknowledgment that publication
    has occurred or constitute a waiver of confidentiality.

    Composition Toolbox software is the proprietary
    and confidential property of Stonehand Inc.

**************************************************************************/

#include "scparagr.h"
#include "sccolumn.h"
#include "scbreak.h"
#include "scstcach.h"
#include "scglobda.h"
#include "scctype.h"
#include "scmem.h"
#include "scset.h"
#include "scspcrec.h"
#include "scstream.h"
#include "sctextli.h"
#include "scfileio.h"
#include "scpubobj.h"
#include "sccallbk.h"
#include "scapptex.h"

#ifdef _RUBI_SUPPORT
#include "scrubi.h"
#endif

/*=====================================================================*/
/* ==================================================================== */

void scContUnit::Read( scSet*        enumTable,
                       APPCtxPtr     ctxPtr,
                       IOFuncPtr     readFunc )
{
    scTBObj::Read( enumTable, ctxPtr, readFunc );

    ReadLong( fParaCount, ctxPtr, readFunc, kIntelOrder );

    fCharArray.Read( ctxPtr, readFunc );

    fSpecRun.Read( ctxPtr, readFunc );

    long diskid;
    ReadLong( diskid, ctxPtr, readFunc, kIntelOrder );

    TypeSpec spec( (stSpec*)APPDiskIDToPointer( ctxPtr, diskid, diskidTypespec ) );
    defspec_ = spec;

#ifdef _RUBI_SUPPORT
    long rubipresent;
    ReadLong( rubipresent, ctxPtr, readFunc, kIntelOrder );
    if ( rubipresent ) {
        AllocRubiArray();
        fRubiArray->Read( ctxPtr, readFunc, rubipresent );
    }
```

```
                    // this should be executed first
    void    append( TypeSpec& );

    void    append( const uchar* str, int );
    void    append( stUnivString& );
    void    append( UCS2 );

    int     get( UCS2&, TypeSpec& );
    void    reset()
            {
                choffset_ = 0;
            }

    TypeSpec&   paraspec()
                {
                    return paraspec_;
                }

        // CAUTION the semantics this are a bit bizarre
    stPara& operator=( const stPara& );

    void    setparaspec( TypeSpec& ts );

    int     validate() const;
    int     complete();

private:
    TypeSpec                paraspec_;
    scSizeableArray<UCS2>   ch_;
    scSpecRun               specs_;
    int32                   choffset_;
};

/*==================================================================== */

#endif /* _H_SCPARAGR */
```

```
                                TypeSpec&, eUnitType& );

    long            ExternalSize( void ) const;

    Bool            FindLocation( long&, Bool&, scTextline*&, MicroPoint&, eContentMovement );

    void            FreeScrap( long& );

    Bool            FindString( const UCS2*, const SearchState&, long&, long& );


    int             FindString( const stUnivString&, const SearchState&, int32, int32, int32& );
    int             ReplaceToken( const stUnivString&, int32, int32& );
    int             GetToken( stUnivString&, int32, int32 ) const;

    Bool            ReplaceWord( CharRecordP&   startChRec,
                                 scSpecRecord*& specRec,
                                 long           startOffset,
                                 long&          endOffset,
                                 long&          limitOffset,
                                 UCS2*          chBuf,
                                 UCS2*          replaceBuf );

        //////////////////////////////////////////////////////////////////
        //////////////////////////////////////////////////////////////////
        //////////////////////////////////////////////////////////////////
protected:

#if SCDEBUG > 1
    eRefEvent       Reformat2( scCOLRefData&, PrevParaData&, int, int&, int& );
#else
    eRefEvent       Reformat2( scCOLRefData&, PrevParaData&, int, int& );
#endif

    Bool            ResetWidow( scCOLRefData&   cData,
                                Bool            testGetStrip );


    scTextline*     fFirstline;     /* firstline of paragraph */

    long            fParaCount;     /* the # of this para in the stream */

    TypeSpec        defspec_;

    scSpecRun       fSpecRun;

    scCharArray     fCharArray;     /* the charArray */

#ifdef _RUBI_SUPPORT
    scRubiArray*    fRubiArray;
#endif
};

inline UCS2 PARACharAtOffset( scContUnit* p, long offset )
                    { return p->GetCharArray().GetCharAtOffset( offset ); }

/* ================================================================== */

#define NEXT_LINE               LONG_MAX
#define PREV_LINE               LONG_MIN

#define PARAFirstInChain(p) ((scContUnit*)p->FirstInChain( ))
#define PARAChSize( p )     ( (p)->GetCharArray().GetContentSize() )


/* ================================================================== */

class stPara {
public:
            stPara();
            stPara( TypeSpec& );

            ~stPara();
    TypeSpec        GetDefaultSpec( void ) const
                        {
                            return defspec_;
                        }

#ifdef _RUBI_SUPPORT
    Bool            GetAnnotation( int nth, long, long, scAnnotation& );
    void            ApplyAnnotation( long, long, const scAnnotation& );

    void            AllocRubiArray( void );
    void            AllocRubiArray( const scRubiArray& );
    void            DeleteRubiArray( void );
    void            CopyRubiArray( const scRubiArray& );

    scRubiArray*    GetRubiArray( void ) const  { return fRubiArray; }
    Bool            IsRubiPresent( size_t, size_t );
#endif

    int             operator==( const scContUnit& ) const;
    int             operator!=( const scContUnit& p2 ) const { return !(*this == p2); }


    scContUnit*     GetPrev( void ) const   { return (scContUnit*)Prev(); }
    scContUnit*     GetNext( void ) const   { return (scContUnit*)Next(); }


    scTextline*     NextColumn( scColumn*& );

    void            SetFirstline( scTextline* firstline )   { fFirstline = firstline; }
    scTextline*     GetFirstline( void ) const              { return fFirstline; }

                        // find the line associated with the indicated offset
    scTextline*     FindLine( long offset ) const;


    scTextline*     GetLastline( void ) const;
    scTextline*     GetLastVisibleLine( void ) const;
    scContUnit*     GetLastVisiblePara( void ) const;
    scContUnit*     GetPrevVisiblePara( void ) const;

                    // this returns the column that this paragraph starts in
    scColumn*       GetFirstCol( void ) const;


    scContUnit*     Earlier( const scContUnit* ) const;

    long            GetCount( void ) const  { return fParaCount; }
    void            SetCount( long cnt )    { fParaCount = cnt; }

                        // do this paragraph and this column intersect
    Bool            ColSect( const scColumn* ) const;

    void            ForceRepaint( long, long );


    void            Renumber( void );
    void            Retabulate( TypeSpec ts );
    void            SelectWord( long, long&, long& );
    void            SetStyle( long, long, TypeSpec, Bool retabulate, Bool forceRepaint );

    scContUnit*     CopyText( long, long ) const;

 #if SCDEBUG > 1
    virtual void    scAssertValid( Bool recurse = true ) const;
    virtual void    DbgPrintInfo( int debugLevel = 0 ) const;
    void            DebugParaSpecs();
 #else
    virtual void    scAssertValid( Bool = true ) const{}
 #endif


    void            ChInfo( long, UCS2&,
                            ulong&, MicroPoint&, MicroPoint&,
    void            Unlink( void );

    void            ReadAPPText( stTextImportExport& );
    void            WriteAPPText( stTextImportExport& );


    long            ReadStream( APPCtxPtr, IOFuncPtr );
    void            WriteStream( Bool, APPCtxPtr, IOFuncPtr );


                        // FILE I/O

                        // complete the read
    virtual void    Read( scSet*, APPCtxPtr, IOFuncPtr );

                        // complete the write
    virtual void    Write( APPCtxPtr, IOFuncPtr );

                        // restore the pointers after completing a read
    virtual void    RestorePointers( scSet* );

    void            PasteText( const scContUnit*, long& );
    Bool            ClearText( long, long );

    scContUnit*     PasteParas( const scContUnit*, long& );


    void            TextTrans( long, long, eChTranType, int );


    TypeSpec        SpecAtEnd( void );
    TypeSpec        SpecAtOffset( long );
    TypeSpec        SpecAtStart( void );


                        // does this para contain this spec
    Bool            ContainTS( TypeSpec );


    void            GetTSList( scTypeSpecList& );
    void            OffsetGetTSList( long, long, scTypeSpecList& );

    void            OffsetGetCharSpecList( long, long, scSpecLocList& );

                    // the chararray is null terminated, so to tell
                    // use how many characters we have we need to subract
                    // one from the array size
                    //
    long            GetContentSize( void ) const
                        {
                            return fCharArray.GetContentSize();
                        }
    scCharArray&    GetCharArray( void )
                        {
                            return fCharArray;
                        }

    scSpecRun&      GetSpecRun( void )
                        {
                            return fSpecRun;
                        }
    const scSpecRun& GetSpecRun( void ) const
                        {
                            return fSpecRun;
                        }

                        void            CopySpecRun( const scSpecRun& );

    void            SetDefaultSpec( TypeSpec& );
    TypeSpec&       GetDefaultSpec( void )
                        {
                            return defspec_;
                        }
                    ~scContUnit();

#if SCDEBUG > 1
    int             fReformatEvent;         // the tick the last time the paragraph
                                                // was reformatted - used only for debugging
    void            SetReformatEvent( int event )   { fReformatEvent = event; }
#endif


    void            Free( scSelection* sel = 0 );

    void            InitParaSpec( TypeSpec& );

    void            LockMem( CharRecordP&, scSpecRecord*& );
    void            UnlockMem( void  );

    scStream*       GetStream( void ) const     { return (scStream*)FirstInChain(); }

    scTextline*     LocateFirstLine( scCOLRefData&,
                                     TypeSpec,
                                     scColumn*&,
                                     MicroPoint&,
                                     scLEADRefData&,
                                     PrevParaData& );

    scContUnit*     KeySplit( long&     offset,
                              scKeyRecord&  keyRec,
                              long&     tmMove,
                              short&        rebreak );
    scContUnit*     Split( long& );
    scContUnit*     Merge( long& );

    void            Insert( const CharRecord&,
                            TypeSpec&,
                            long );


    scContUnit*     KeyInsert( long&,
                               scKeyRecord&,
                               long&,
                               short&,
                               Bool,
                               TypeSpec,
                               Bool& );

    void            CharInsert( long,
                                long&,
                                scKeyRecord&,
                                long&,
                                short&,
                                Bool,
                                TypeSpec );


    scContUnit*     Copy( scContUnit* ) const;


    void            Iter( SubstituteFunc, long, long& );

    virtual void    Mark( const scLogBits& bits );

#if SCDEBUG > 1
    eRefEvent       Reformat( scCOLRefData&, PrevParaData&, int, int& );
#else
    eRefEvent       Reformat( scCOLRefData&, PrevParaData&, int );
#endif

    void            Deformat( void );

    void            PostInsert( scContUnit* );

    void            Append( scContUnit* );
```